STATE VARIABLE HARMONIC BALANCE ANALYSIS OF NONLINEAR CIRCUITS BASED ON WAVES

by

Ryan Gregory Bruce Plater

A Thesis
Presented to Lakehead University
in Partial Fulfilment of the Requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering

Thunder Bay, Ontario, Canada
September 17, 2010

Abstract

Circuit simulation involving nonlinear elements can be a challenging task. From these challenges rises a demand for finding better and more efficient ways of solving such problems. This work provides a novel approach for performing harmonic balance (HB) analysis using the fREEDA circuit simulator. The proposed method is an extension of the method of multiple reflections for multiple ports. In addition the method is formulated in terms of power waves and state variables at the nonlinear devices. The HB problem is then solved using a procedure which resembles the signal propagation within the actual circuit. This method could be efficiently parallelized since it does not require a large matrix decompositions at each iteration.

Several approaches to improve convergence properties are investigated. The first involves adding capacitors in parallel with the nonlinear device ports, this allows the fixed-point iterations to always be convergent. These capacitors are only active in a separate time dimension and do not affect the steady-state solution. The harmonic balance solution is found when the transient response in this time dimension is extinguished. Another strategy to improve convergence is the combination of fixed-point iterations with the gradient descent method. The effect of a vector extrapolation method to accelerate convergence is also investigated. Simulation results for various strongly nonlinear circuits is presented.

This thesis covers the background of harmonic balance analysis, literature review, derivation of the proposed method, improvements, preliminary results, as well as future work.

Acknowledgements

I would like thank my supervisor Dr. Carlos E. Christoffersen for the tremendous support and guidance throughout my graduate studies at Lakehead University. It is my privilege studying with him and being a part of his research group.

I would also like to express my sincere thank you to the remaining faculty and staff of the faculty of engineering for their support and teaching over the course of my studies. Another thank you to my peers, colleagues, friends and family for their encouragement and support throughout this process. Finally a special thank you to Muhammad Kabir for his help during my research work.

Ryan Gregory Bruce Plater

rplater@lakeheadu.ca

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol	Meaning
$\bar{A}$	Transmitted Power Wave Vector
$\bar{A}_0$	Source Power Vector
$\bar{B}$	Reflected Power Wave Vector
$\bar{c}_k$	Extrapolation Coefficient Vector
d	Device Number
$[D]$	Diagonal Matrix Containing $\sqrt{Z_{ref}}$ in the diagonal elements
$\bar{f}_i$	Current Error Vector
$\bar{f}_v$	Voltage Error Vector
G_1	Parallel Conductance Value for 'Pseudo Transient'
∇H	Gradient of H
i_L	Time Domain Linear Port Current
I_L	Frequency Domain Linear Port Current
$\bar{i}_{NL}$	Time Domain Nonlinear Port Current
$\bar{I}_{NL}$	Frequency Domain Nonlinear Port Current
$[J_i]$	Current Jacobian Matrix
$[J_v]$	Voltage Jacobian Matrix
k	Iteration Number
m	Harmonic Number
$[M_{SV}]$	Frequency Domain Modified Nodal Admittance Matrix
n	State Number
p	Port Number
p_n	Convergence Parameter
q	Time Domain Charge
Q	Frequency Domain Charge
$[S]$	Scattering Parameter Matrix
$\bar{S}_{SV}$	Frequency Domain Source Vector
$\bar{u}_k$	Extrapolation Difference Vector
$[U_k]$	Extrapolation Difference Matrix
v_L	Time Domain Linear Port Voltage
V_L	Frequency Domain Linear Port Voltage
$\bar{v}_{NL}$	Time Domain Nonlinear Port Voltage
$\bar{V}_{NL}$	Frequency Domain Nonlinear Port Voltage
$\hat{x}$	Estimate of Solution
$\bar{x}$	Time Domain State Variable Vector
$\bar{x}_D$	Time delay State Variable
$\bar{X}$	Frequency Domain State Variable Vector
$[Y]$	Admittance Matrix
Z_{ref}	Reference Transmission Line Characteristic Impedance

$\mathcal{F}$	Fourier Transform
$\mathcal{F}^{-1}$	Inverse Fourier Transform
$\Re$	Real Part
$\Im$	Imaginary Part
Γ	Reflection Coefficient
α	Nonlinear Port Newton Update Scaling Factor
ω	Frequency (rad/s)
Ω	Diagonal ω Matrix
Φ	Phase
τ_i	Time Delay

List of Abbreviations

Abbreviation	Meaning
AC	Alternating Current
DC	Direct Current
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
GD	Gradient Descent
GMRES	General Minimum Residual
HB	Harmonic Balance
INHB	Inexact Harmonic Balance
LAPACK	Linear Algebra PACKage
LU	Lower Upper (Matrix Decomposition)
LNA	Low Noise Amplifier
MESFET	Metal Semiconductor Field Effect Transistor
MMIC	Monolithic Microwave Integrated Circuit
MPE	Minimum Polynomial Extrapolation
MTL	Matrix Template Library
NL	Nonlinear
NA	Not Applicable
RMS	Root Mean Squared
SVHB	State Variable Harmonic Balance
Tol Step	Tolerance Stepping

Chapter 1

Introduction

1.1 Motivation and Objectives of This Study

As technologies advance new possibilities emerge. With these advances the demand for faster, smaller and more complex designs occur. The electronics field is not exempt from this and may be one of the major driving forces in technology. From this stems the need for quicker and more efficient ways to design and simulate electronic circuit operation. There are two common approaches used for solving such problems. They are commonly divided into two broad groups being time and frequency based. There are various available approaches for implementing these two methods.

This thesis presents a harmonic balance (HB) approach using a relaxation technique based on the work performed by Kerr [3] and Hicks and Khan [4]. The idea draws from the multiple reflection technique of Kerr and considers a network partition similar to the approach proposed by Hicks and Khan. Transmission line theory and power waves are used to model the circuit operation along with various techniques to improve and control convergence. A rigorous extension using waves (unlike the extensions developed in References [9,12]) for solving harmonic balance analysis formulated using state variables for nonlinear circuit simulations is presented for the first time. Combining the use of power waves with the addition of parallel connected capacitors, extrapolation and gradient descent methods are also tested for the first time.

The basic existing harmonic balance techniques (discussed in Chapter 2) work well for circuits of smaller size and that demonstrate fairly linear characteristics. Krylov-subspace methods with inexact Newton harmonic balance using generalized minimum residual technique (GMRES) and preconditioning has demonstrated success for larger nonlinear circuits [2,5]. However there are limitations to these approaches in the size of problem which can be solved, convergence to the solution and CPU time. One of the major limitations of the harmonic balance approaches is convergence. Predicting convergence at the on set of a simulation can not be determined. A matrix preconditioner is used to improve the properties and initial guess of solving a system of equations. The use of a preconditioner can become a requirement for successful convergence and thus plays a crucial role in obtaining a successful solution [2]. These improvements still do not entirely overcome all of these limitations.

The proposal of using wave quantities, more specifically power waves, shows some promise. The use of waves closely represents the actual signal propagation during circuit operation. Iterative methods based on power waves can be made theoretically constrained to avoid the possibility of divergence.

The majority of techniques mentioned in Chapter 2 have been formulated using Kirchoff's voltage law and/or Kirchoff's current law to represent the circuit quantities. The proposed process is based on transmission line theory and the propagation of wave quantities, by using transmission lines to connect the linear and nonlinear sub-circuits together. The transmission lines have been selected to have a length of zero and thus have no effect on the circuit. The use of wave properties should allow for better convergence, the ability to easily divide the problem into smaller pieces and shows promise for parallelization which can take advantage of todays

multi-core and cluster computers.

1.2 Thesis Overview

This thesis covers the following topics. Chapter 2 begins with a brief introduction to the history of solving nonlinear circuits, followed by a review of basic concepts related to circuit simulation. The Harmonic balance concept is then introduced. Various present harmonic balance approaches are described. Finally the circuit simulator used to implement the techniques developed in the thesis is discussed.

Chapter 3 presents a new harmonic balance based technique based on waves. The main equations are derived first, followed by an analysis of convergence and some strategies to improve/accelerate convergence. Numerical simulations are presented in Chapter 4. The performance of the proposed methods is evaluated with the simulations of several circuits: Resistor Diode, Full Wave Rectifier, Charge Pump, MESFET Amplifier, and a Soliton line. Chapter 5 presents conclusions and suggested future research directions.

Chapter 2

Literature Review

2.1 Introduction

There are two main categories that most circuit simulation techniques fall into, those being time and frequency based. Time based circuit simulations such as transient analysis use small time steps to sample the circuit operation [2]. The use of frequency based methods implies that there is some advantage over time based methods for certain circuit solutions. One of such frequency based methods is harmonic balance.

Unlike transient analysis, harmonic balance calculates the steady state solution directly. This is accomplished by using a linear addition of sinusoids to produce the solution [1]. There have been many different approaches proposed to solve circuits using the harmonic balance technique. The conventional process used for harmonic balance is to divide the circuit up into two subnetworks comprised of the linear and nonlinear elements of the circuit [2]. The linear network can then be solved in the frequency domain and the nonlinear network solved for in the time domain. This being the reason that harmonic balance is sometimes referred to as a mixed domain method [1]. The name harmonic balance stems from the process where currents are balanced between the linear and non-linear sub-circuits [4]. The linear network is typically represented in the form of an admittance matrix (Y-parameters) [2]. Other representations such as the scattering matrix (S-parameters) have been proposed and show some promise with certain techniques [5,17].

As the number of elements in the circuit increase so does the number of unknowns and therefore the complexity in solving the system. There have been different proposed methods to help deal with these complexities when solving a system of equations. Details of some of these approaches will be discussed in the following sections. Section 2.2 presents numerical techniques that are used to solve the harmonic balance equations, such as Newton's method. The use of extrapolation and gradient descent techniques is also covered. The basics of voltage and power waves is discussed in Section 2.3. Section 2.4 discusses conventional and current harmonic balance techniques and various approaches to solve the harmonic balance equations. The solution approaches discussed included, optimization, relaxation techniques, Newton based methods, and the use of matrix preconditioners. Section 2.5 discusses the architecture of the fREEDA simulator and implementation of a new simulation technique.

2.2 Numerical Techniques

2.2.1 Newton's Method

Newton's method more correctly referred to as the Newton-Raphson method is one of the most common numerical approaches used in todays computers to solve systems of nonlinear equations. It is especially powerful when it comes to solving complex systems with nonlinearities where conventional substitution methods are not possible or difficult [7]. Practically all of todays circuit simulation programs use some form of Newton based technique. Newton's method extrapolates to the axis of the independent variable using its first derivative [2].

Consider the system of nonlinear equations given by:

$$\begin{aligned} f_1(x_1, x_2, .., x_n) &= 0 \\ f_2(x_1, x_2, .., x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, .., x_n) &= 0 \end{aligned} \tag{2.1}$$

If the functions $(f_1, f_2, .., f_n)$ are expanded using a Taylor series about an arbitrary point $(x_1, x_2, .., x_n)$ and only the linear terms retained we can obtain exact solution values $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ [7]. Setting the equations defined from the Taylor series expansion equal to zero and defining a Δx_i term as the correction to the guess of x_i,

$$\Delta x_i = \hat{x}_i - x_i \quad where \quad (i = 1,2,\ldots, n) \quad , \tag{2.2}$$

can be formed. Rewriting the the Taylor series expansion in general form,

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \vdots \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \\ \vdots \\ \vdots \\ -f_n \end{bmatrix} . \tag{2.3}$$

The matrix on the left of Equation (2.3) is referred to as the Jacobian and is calculated for each approximation of x_i. The use of the first order partial derivatives for each device and harmonic considered creates the Jacobian matrix. The Jacobian contains the maximum amount of information about the system and its error. With this large amount of information about the system good convergence characteristics can typically be achieved [2].

The Δx_i term is solved at each iteration (k) and the approximation of the solution x_i is updated by,

$$x_i^{k+1} = x_i^k + \Delta x_i^k \tag{2.4}$$

This process can be broken down into six main steps. First the initial guess is selected (x_i^0), the function is evaluated at x_i^0 and the partial derivatives are calculated to form the Jacobian matrix. The update Δx_i is solved, a new solution x_i^1 is determined and checked for convergence to a solution. If convergence to a solution was achieved then the process is stopped otherwise it repeats using the newest approximate solution as the initial guess [7].

Newton's method is quite good at solving systems of this form. However this approach still does not guarantee convergence if the the initial guess is far from the solution. It also becomes impractical for solving very large systems of equations. This occurs because the Jacobian size becomes too large to directly factor efficiently, even in today's powerful computers [15]. These issues and limitations lead to modifications of Newton's method to help alleviate them as discussed in Section 2.3.

2.2.2 Minimum Polynomial Extrapolation

Minimum polynomial extrapolation (MPE) is a technique to find the fixed point (s) of a sequence of vectors which satisfies the system of equations. If the system in question is linear using a MPE will extrapolate to the fixed point. For systems which are nonlinear performing a MPE with enough samples has the ability to extrapolate to a solution relatively close to the correct one. Minimum polynomial extrapolation is based on using vector differences to calculate the solution [19].

Given a vector sequence $\bar{x}_{1,}\bar{x}_{2,}\cdots,\bar{x}_{k+1}$ generated by an equation of the form:

$$\bar{x}_{j+1} = [A]\bar{x}_j + \bar{b}, \qquad j=0,1,2,\cdots, \tag{2.5}$$

Where $[A]$ is a fixed matrix and $\bar{b}$ a fixed vector. Both $[A]$ and $\bar{b}$ do not necessarily have to be known, only that there is a way of generating $\bar{x}_{1,}\bar{x}_{2,}\cdots,\bar{x}_{k+1}$ [19]. From an initial starting point of $\bar{x}_0$ a sequence $\bar{x}_{1,}\bar{x}_{2,}\cdots,\bar{x}_{k+1}$ is generated. Using a fixed amount of samples k, a $N\times k$ matrix is formed, where the columns are the vectors of the vector differences:

$$\bar{u}_j = \Delta\bar{x}_j = \bar{x}_{j+1} - \bar{x}_j \tag{2.6}$$

$$[U]\equiv[U_k] = [\bar{u}_0,\bar{u}_1,\cdots,\bar{u}_{k-1}] \tag{2.7}$$

Using the $[U]$ matrix a coefficient c vector is defined to satisfy.

$$[U]\bar{c} = -\bar{u}_k \tag{2.8}$$

The values of $\bar{c}$ are calculated using the following formula,

$$\bar{c} = -[U]^{+}\bar{u}_k \ , \tag{2.9}$$

where $[U]^{+}$ is the Moore Penrose pseudo-inverse.

For any consecutive term sequence with a length of $k+1$ the fixed point (s) can be calculated [19] as follows,

$$\sum_{j=0}^{k} c_j x_{m+j} = \left(\sum_{j=0}^{k} c_j\right) s \tag{2.10}$$

with the m term being an offset within the generated sequence and $c_k = 1$. The conditions for a

successful extrapolation calculation requires that the fixed point exists and the number of samples k is equal or greater then the number of nonzero eigenvalues of the $[U]$ matrix.

The outcome of using MPE on nonlinear equations can be incorrect if the selected sequence length k is too short, the generated sequence is far from the correct solution or the system of equations exhibits strong nonlinearity.

2.2.3 Gradient Descent

Gradient descent is a first order optimization routine which can be used to find the local minima scalar field. During an iteration routine the gradient of the function is calculated and used to determine the direction of the update. The update is scaled toward the minimum of the function or proportionally to the negative of its gradient. Analogous to the descent method is gradient ascent which is used to find local maxima. Care must be taken when using gradient methods. Performing a gradient update on the system before being relatively close to solution can trap the iteration process at an incorrect result.

Given a function $F(x^k)$ where at solution equals zero. The gradient of the function can be calculated if the function is defined and differentiable in the area around a selected solution point g [31]. The fastest way to decrease $F(x^k)$ from a starting point of g is to move in the direction of the negative gradient or $-\nabla F(g)$. If

$$h = g - \gamma \nabla F(g) \ , \tag{2.11}$$

for a value of γ small enough, then $F(g) \geq F(h)$. From this observation and starting from an initial guess of x^0 for a local minimum of the function $F(x^k)$. A generated sequence $x^{0,} x^{1,} x^{2,} \ldots$ can be calculated from:

$$x^{k+1} = x^k - \gamma^k \nabla F(x^k) \quad , \quad k \geq 0 \quad , \tag{2.12}$$

such that $F(x^0) \geq F(x^1) \geq F(x^2) \geq \ldots$ which will hopefully converge to the local minimum. The value of γ is not necessarily the same for each iteration.

2.3 Basics of Waves

Consider the lossless transmission line in Fig. 2.1 with characteristic impedance Z_{ref}, terminating load impedance Z_L and sinusoidal voltage source E_S.

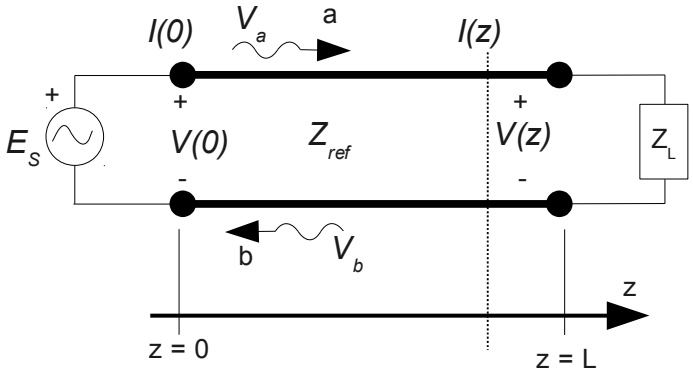

Fig. 2.1 Lossless Transmission Line

The sinusoidal wave produced by E_S and propagating in the positive z direction is reflected back in the negative z direction when it reaches the termination at $z = L$. The voltage and currents at any point along the transmission line consists of both positive and negative going waves. The voltage and current at any point along the transmission line consists of waves propagating along the positive and negative z direction:

$$V(z) = V_{a'} e^{-j\beta z} + V_{b'} e^{+j\beta z} \tag{2.13}$$

$$I(z) = \frac{V_{a'}}{Z_{ref}} e^{-j\beta z} - \frac{V_{b'}}{Z_{ref}} e^{+j\beta z} \tag{2.14}$$

The magnitudes of voltage in the positive and negative z direction are $V_{a'}$ and $V_{b'}$ respectively and β is the phase constant [8]. The ratio between the positive z direction propagating wave and the negative z direction propagating wave is referred to as the reflection coefficient Γ and is given by,

$$\begin{aligned} \Gamma &= \frac{V_{b'} e^{+j\beta z}}{V_{a'} e^{-j\beta z}} \\ \Gamma &= \frac{Z_L - Z_{ref}}{Z_L + Z_{ref}} \end{aligned} \tag{2.15}$$

With V_a and V_b being the RMS values of traveling wave voltage, then $|V_b| = \frac{|V_{b'}|}{\sqrt{2}}$ and $|V_a| = \frac{|V_{a'}|}{\sqrt{2}}$. The power flowing in the positive z direction is given by,

$$P = \frac{|V_a|^2}{Z_{ref}} - \frac{|V_b|^2}{Z_{ref}} \tag{2.16}$$

2.3.1 Voltage Wave

Given the length of the transmission line is L then the voltage and current at $z = L$ is,

$$\begin{aligned} V(L) &= V_{a'} e^{-j\beta L} + V_{b'} e^{+j\beta L} \\ I(L) &= \frac{V_{a'}}{Z_{ref}} e^{-j\beta L} - \frac{V_{b'}}{Z_{ref}} e^{+j\beta L} \end{aligned} \tag{2.17}$$

Setting the length of the transmission line equal to zero the voltage and current become,

$$\begin{aligned} V &= V_{a'} + V_{b'} \\ &= \sqrt{2}(V_a + V_b) \end{aligned} \tag{2.18}$$

and

$$I = \frac{V_{a'}}{Z_{ref}} - \frac{V_{b'}}{Z_{ref}} = \frac{\sqrt{2}}{Z_{ref}}(V_a - V_b) \tag{2.19}$$

The use of a zero length transmission line allows the voltage and currents at any node to be converted to voltage waves. The use of voltage waves can then be used as parameters to formulate circuit analysis [8].

2.3.2 Power Wave

The use of waves is not limited to just voltage. The circuit parameters can also be formulated using power waves. Considering ' a ' as the positive propagating power wave and ' b ' as the negative propagating power wave in such a away that $|a|^2 = \frac{|V_a|^2}{Z_{ref}}$ and $|b|^2 = \frac{|V_b|^2}{Z_{ref}}$ in (Fig. 2.1) [8]. The voltage and currents can be expressed as,

$$V = \sqrt{Z_{ref}}(a + b) \tag{2.20}$$

and

$$I = \frac{1}{\sqrt{Z_{ref}}}(a - b) \tag{2.21}$$

The power wave equations (2.20) and (2.21) are the bases used throughout this work.

2.4 Harmonic Balance Formulation

Early use of piecewise harmonic balance approach is demonstrated by Nakhla and Vlach [6]. The basis of piecewise harmonic balance is to separate the linear and nonlinear devices into two sub networks. Current and voltage equations are formed for the two ports. An error function

is formed by taking the difference between the currents and/or the voltages of the linear and nonlinear ports. Detailed equation formulation is presented next.

2.4.1 Conventional Harmonic Balance Equations

As mentioned above the idea is to partition the circuit into its linear and nonlinear components. This reduction allows the nonlinear sub-circuit to be solved independently of the linear sub-circuit. The linear network can then be solved directly in the frequency domain. While the nonlinear subnetwork element models can be solved in the time domain and subsequently converted to the frequency domain. This partitioning can be seen in Fig. 2.2.

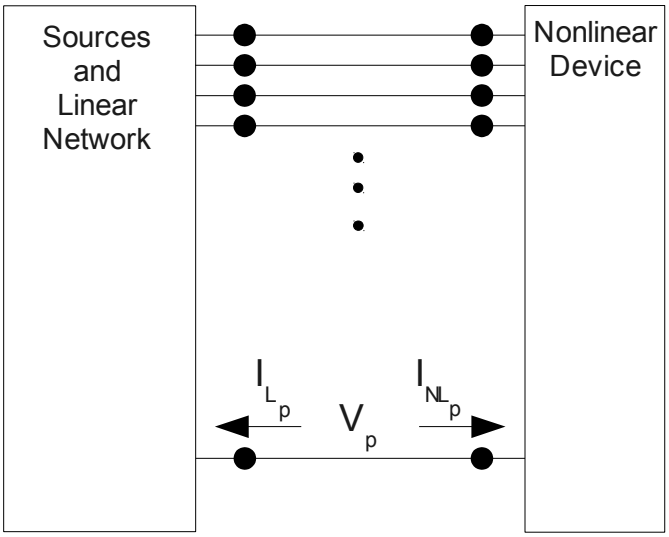

Fig. 2.2 Harmonic Balance General Partitioning

The voltage and currents at each port (p) are assumed to be periodic having a period of T which is defined as $T = \frac{2\pi}{\omega}$ and ω being the angular velocity. The voltage at port p can be expressed using the following equation:

$$v_p(t) = \Re\left\{\sum_{k=0}^{m} V_k e^{(j\omega kt + \phi_k)}\right\} \quad , \tag{2.22}$$

where the unknown values are the amplitude V_k and phase ϕ_k for each frequency.

The error function is formulated as the sum of the currents at the ports connecting the linear and nonlinear ports [2].

$$F(V) = I(V) + j\,\Omega\,Q(V) + YV + I_S = 0 \quad , \tag{2.23}$$

where $I(V)$ is the current from the nonlinear conductances or voltage-controlled sources and $\Omega Q(V)$ is the current contribution from the nonlinear capacitors. These two terms together represent the current contribution by the nonlinear network (I_{NL}). The Ω symbol represents a diagonal matrix containing $k\,\omega$ elements in the main diagonal,

$$\Omega = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 0 & \omega & \ddots & \cdots & 0 \\ \vdots & \vdots & 2\omega & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & m\,\omega \end{bmatrix} \tag{2.24}$$

The I_S term is the current contribution from the circuit sources and YV is the current contribution from the linear elements. These two together represent the linear port current I_{LIN}. The name of harmonic balance is easily seen because a correct guess for V in Equation (2.23) will balance the linear and nonlinear currents for each Harmonic.

Consider just the nonlinear part from Equation (2.23). The nonlinear current contribution I_{NL} can be written as:

$$I_{NL}(V)=\mathcal{F}\{\boldsymbol{i}[\mathcal{F}^{-1}(V)]\}+\Omega\mathcal{F}\{\boldsymbol{q}[\mathcal{F}^{-1}(V)]\} \tag{2.25}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and Inverse Fourier transforms and $i()$ and $q()$ represent the nonlinear currents and charge, respectively, in the time domain [2]. The estimate of voltage (V) in the frequency domain is converted into the time domain using Fourier transform. The nonlinear current i_{NL} is then solved for in the time domain. Using inverse Fourier transform i_{NL} is then converted back to the frequency domain and used to calculate the error $F(V)$ [2].

For the correct selection of V in Equation (2.14) $F(V)$ will be approximately zero. Due to the iterative technique used to solve such a problem, a solution is found when all elements of the $F(V)$ vector are less than a defined tolerance.

A simple harmonic balance example can be demonstrated using the circuit in Fig. 2.3. The source is made up of both a DC voltage and an AC fundamental frequency. First defining the linear current (I_{LIN}), nonlinear current (I_{NL}) and port voltage allows for easy equation setup.

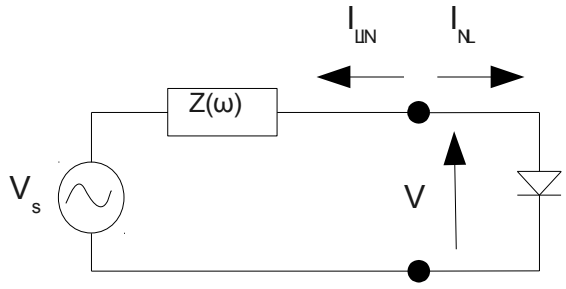

Fig. 2.3 Simple Diode Circuit

The circuit is divided up into its two sub-circuits. Fig. 2.4 represents the linear sub-circuit in the frequency domain. The diode is replaced with the voltage seen at the terminals.

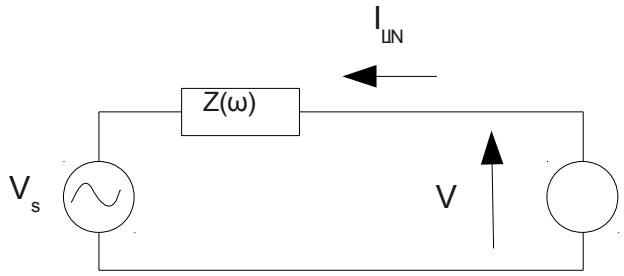

Fig. 2.4 Linear Sub-Circuit (I_{LIN})

Writing the equation for the linear current in the frequency domain Equation (2.26).

$$I_{LIN} = \frac{V(k\omega_p) - V_s(k\omega_p)}{Z(k\omega_p)} \tag{2.26}$$

Fig. 2.5 represents the nonlinear sub-circuit in the time domain.

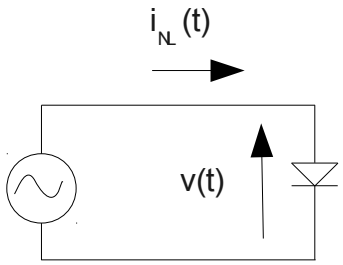

Fig. 2.5 Nonlinear Sub-Circuit (I_{NL})

Writing the equation for the nonlinear current in the frequency domain is accomplished by first calculating the diode current in the time domain using,

$$i_{NL}(t) = i_{sat}(e^{\frac{v_{NL}(t)}{nv_T}} - 1) \tag{2.27}$$

where i_{sat} is the saturation current, v_T is the thermal voltage and n is the ideality factor of the diode. The error function for the circuit is given by,

$$I_{LIN}(k\omega_p) + I_{NL}(k\omega_p) = 0 \tag{2.28}$$

Substituting the two currents into the error function Equation (2.28) and using Fourier transform to convert the nonlinear current into the frequency domain,

$$\frac{V(k\omega_p)-V_s(k\omega_p)}{Z(k\omega_p)} + \mathcal{F}[I_{sat}(e^{\frac{\mathcal{F}^{-1}(V(k\omega_p))}{nv_t}}-1)] = 0 \qquad (2.29)$$

is formed. A correct guess of $V(k\omega_p)$ will balance the linear and nonlinear currents for each harmonic therefore satisfying Equation (2.29). The diode example is a brief demonstration using the harmonic balance approach. The more complex problem of how to efficiently select and update V using the numerical techniques from Section 2.2 is now discussed.

2.4.2 Existing Harmonic Balance Techniques

With the general structure of harmonic balance discussed lets look further into how the solution is found. The task that is being attempted is basically finding the zeros of the system of equations. Thankfully these problems have been largely studied by mathematicians. The most common approach involves using Newton method [2,5,14,15]. Newton's method is a very powerful and commonly used technique for indirectly solving complicated problems. However there are still some limitations to this approach. Convergence can not be guaranteed and the size of the Jacobian matrix produced using Newton's method is directly related to the number of nonlinear devices and harmonics considered. These two major limiting factors can lead to problems finding the solution of strongly nonlinear circuits and inefficiencies for large circuits due to long solution times and large amounts of computer memory usage. To overcome some of these limitations, modifications and alternative techniques have been proposed and used. Some of which are, Optimization [2], Relaxation methods [3,4], matrix conditioning [2], Sparse Matrix

Solvers incorporating Krylov-Subspace methods with preconditioners [5,11,14,15].

2.4.3 Optimization

Optimization routines can be used to solve equation (2.30) using a norm of the error(ϵ).

$$\epsilon = F^*(V)F(V) \tag{2.30}$$

where $F^*(V)$ denotes the complex conjugate transpose of $F(V)$.

This method has an advantage that many scientific libraries already have an optimization routine built in. However this process destroys information about the contribution of each individual element to the error therefore convergence can suffer [2]. This technique is only practical for relatively small and simple circuits. This technique reduces programing complexity and therefore implementation time at the cost of sometimes poor convergence and inefficiency.

2.4.4 Relaxation Techniques

Relaxation methods derive their name from the fact that they allow the iteration process to gradually move towards a solution. One of the simplest examples of relaxation is the bisection method [7]. One main advantage of relaxation techniques are their ease of implementation [2]. These methods show promise on some circuits but in a more general sense tend to lead to poor and unpredictable convergence. Two of the more commonly known methods are those of Kerr [3] and Hicks and Khan [4].

Kerr's method starts by connecting a transmission line with characteristic impedance Z_0 between a linear network port and one nonlinear device, in this case a diode. Equations are

formed by separating the circuit at both ends of the transmission line. This is accomplished by increasing the transmission line length (L) so that the period between transient reflections is long enough that the ends can reach their steady state condition [3]. To solve the circuit the left propagating wave (E_L) is calculated and after a time delay will reach the left side embedding network (Fig. 2.6). The right propagating wave (E_R) is calculated and after another time delay will reach the right side diode. The multiple reflections continue until the voltage and current that are calculated at each end of the transmission line are equal [3].

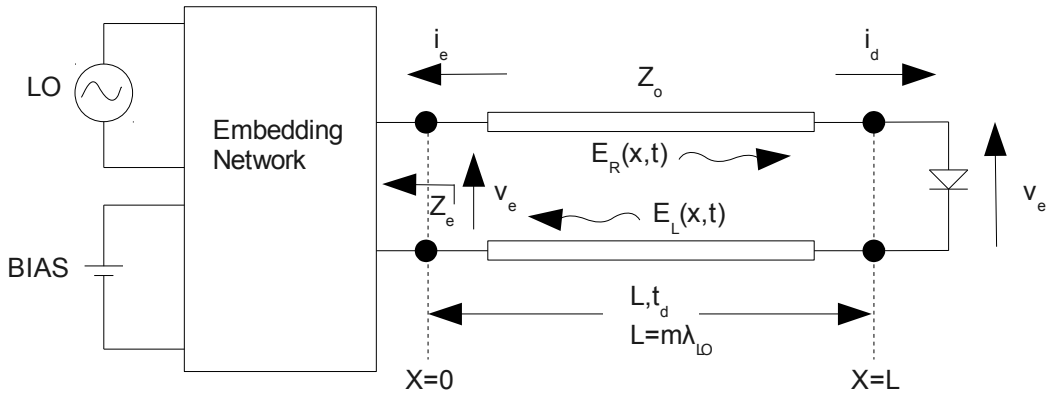

Fig. 2.6 Kerr's Circuit Partitioning

Hicks and Khan propose a method using either voltage or current updates with a circuit partitioning as seen in Fig. 2.7 The process involves selecting a value of the nonlinear port voltage $v_N(t)$ and using it to determine the nonlinear current $i_N(t)$ [4].

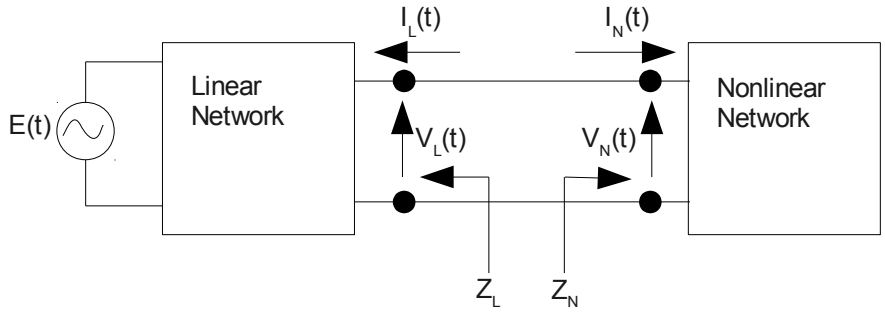

Fig. 2.7 Hicks and Khan Circuit Partitioning

From the circuit topology the nonlinear current must be equal and opposite to the linear current.

$$i_N(t) = -i_L(t) \tag{2.31}$$

The linear current is converted to the frequency domain to obtain $I_L(\omega)$, from which $V_L(\omega)$ can be easily calculated and converted back to the time domain for comparison. Using the calculated value for $v_L(t)$ the error between $v_L(t)$ and $v_N(t)$ can be determined. The iteration process is continued until [4]:

$$v_N^{k+1}(t) = v_N^k(t) = v_L^k(t) \tag{2.32}$$

The voltage update is calculated using:

$$v_{N(m)}^{k+1} = p_n v_{L(m)}^k + (1-p_m) v_{N(m)}^k \ , \tag{2.33}$$

where m is the harmonic number and k is the iteration number. The p_m term is referred to as the convergence parameter and is bound between the range of $0 < p_m < 1$ [4]. Adjusting the value of p_m can increase the area of convergence which is typically offset by a longer solution time [4]. The equations for the current update method are analogous of those of the voltage method and therefore are not shown.

The use of an "identity network" has also been proposed to help with convergence. An example of an identity network is shown in Fig. 2.8. Where Fig. 2.8(a) is used for the voltage update method and Fig. 2.8(b) for the current update method [4].

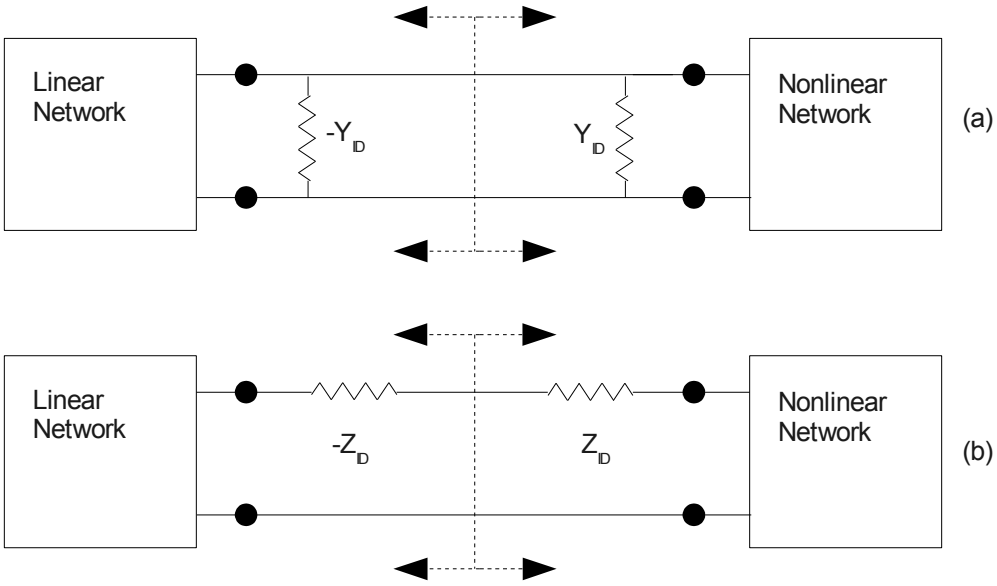

Fig. 2.8 Identity Networks (a)Voltage Update Method (b) Current Update Method

The purpose of the identity network is to modify the harmonic impedance ratios and therefore help with solution convergence [4].

A modified version of Kerr's method [3] is proposed by Tait [12]. The technique is referred to as the Accelerated Fixed Point Algorithm. The approach maintains the fictitious transmission lines between the linear and nonlinear devices and uses the idea of multiple reflections. A Steffensen acceleration technique [30] with a voltage update method similar to that of Hicks and Khan [4] is formulated for the iteration process. For Tait's example of a frequency multiplying circuit the Steffensen acceleration process demonstrates a super linear convergence rate that approaches the quadratic rate of Newton's method [12]. This is accomplished without the need of a derivative calculation [12].

Borich [9] proposes a method based from the work of Tait [12]. The method expands Tait's approach to a more general case supporting any number of nonlinear devices. Complex characteristic impedance values for the fictitious transmission lines are also examined. The use

of complex characteristic impedance values is shown to improve convergence especially in cases with nonlinear capacitances [9].

An advantage to the use of relaxation techniques is their somewhat easier implementation [2]. They however often demonstrate poor convergence performance in general terms and difficulties arise when dealing with large circuits. The general convergence properties of Borich's work is shown by Blakey to be possible for circuits containing only a few nonlinear devices and recommends the use of conventional harmonic balance for more complex circuits [13]. Improvements in dealing with large circuits and solution convergence have been shown with the use of Newtons method [5,14,15].

2.4.5 Newton Based Methods for Harmonic Balance

As has been discussed before issues arise when the problem size is increased. Krylov subspace with inexact harmonic balance and GMRES (Section 2.4.5) has shown good performance for solving complex systems with success. However large circuits which are strongly nonlinear therefore requiring many harmonics increase this process into a huge computational task. For this reason it becomes apparent that sub dividing the circuit into smaller sub circuits is appealing. This is accomplished by decoupling parts of the circuit that have little to no dependence on one another [15,18]. The technique requires the individual partitions to converge to solution. The entire system then must satisfy global convergence between all sub circuits. The use of Newtons method has been demonstrated to be a powerful tool for solving nonlinear circuits using the harmonic balance approach [2,5,15].

The Jacobian matrix in harmonic balance is made up of partial derivatives of every port

with respect to every frequency and because all harmonics except DC are complex the size must be doubled. The size of the Jacobian matrix required for the full newton method is M x M where, $M = p \times (2m+1)$, p is the number of ports and m is the number of harmonics.

The size of the Jacobian itself becomes an issue in memory storage but the major problem arises when the calculation to factor it occurs. The Jacobian matrix formed during calculations, is not usually sparse. Direct matrix solves such as LU decomposition require a large amount of memory space. The approximate time required to compute the LU decomposition of a dense matrix is related to its dimension cubed. For example doubling the matrix dimension will require approximately eight times the amount of time to decompose [2].

Sparse matrix solvers are used to improve processing time and memory usage as the zero matrix entries are not stored in memory. For sparse matrix solvers to have a benefit the Jacobian matrix is typically pruned to further increase the number of zero entries. As the matrix is factored during calculations the zeros entries can tend to "fill-in" with non-zero values. Care must be taken to prevent or limit such "fill-ins" as they reduce the sparse matrix solving benefits [2].

2.4.6 Krylov-Subspace Methods

The general consensus for solving harmonic balance problems is using a version of Krylov-Subspace method called generalized minimum residual technique or GMRES [2,5]. The technique works the way it sounds. Assuming we must solve an equation of the form $[A]\bar{x} = \bar{b}$ where $[A]$ is a matrix and $\bar{x}$ and $\bar{b}$ are vectors. Defining the residual $\bar{r}$ as $\bar{r} = \bar{b} - [A]\hat{x}$ and using iterative methods to reduce the residual, with the $\hat{x}$ term being an estimation of the solution. However if the matrices are ill-conditioned to begin with, convergence

has very little chance of being achieved [2]. Also for this process to succeed the approximate guess $\hat{x}$ must be relatively close to solution, therefore the use of preconditioning becomes a requirement [2].

2.4.7 Preconditioning

With the techniques discussed in the previous section the use of preconditioning is a requirement for achieving convergence to solution. Some variations of preconditioning are block diagonal [5], static time, iterative time [11] and incomplete LU factorization [15,16].

Block diagonal preconditioning has shown good convergence properties when used with Krylov subspace using inexact harmonic balance and GMRES. As is the problem with full Newton iterations computer storage and computational cost of large circuits limit its efficiency [5]. Rizzoli proposes a method called incomplete LU factorization [15]. The idea is based on exploiting the sparsity pattern in the Jacobian matrix. This sparsity is artificially generated by examining the interaction or dependence between different nonlinear devices [5]. It has been demonstrated that this approach works better when the linear network is represented by its scattering parameters instead of conventional admittance parameters [5,17].

The time based preconditioners both static and iterative are developed using the time counterpart of the frequency domain Jacobian [11]. The difference between static and iterative preconditioners is in the steps of achieving the preconditioning matrix. Terms are dropped in the static method to eliminate the need to iteratively solve for the preconditioning matrix. The iterative method does not neglect these terms [11].

The use of a preconditioner is a requirement for the use of Krylov subspace methods. The

type of preconditioner selected plays an integral role in finding the solution and therefore the convergence performance.

2.5 The fREEDA Circuit Simulator

2.5.1 fREEDA State Variable Formulation

The fREEDA simulator is a free program which supports different circuit analysis types such as transient, harmonic balance and wavelet-based transient analysis. fREEDA is a circuit simulator which uses state variables to represent the nonlinear device models.

fREEDA uses the previously discussed linear and nonlinear sub-circuit partitioning for harmonic balance as shown in Fig. 2.2. The state variable approach models the nonlinear devices using a set of general parametric equations,

$$\bar{i}_{NL}(t) = w\left[\bar{x}(t), \frac{d\bar{x}}{dt}, \cdots, \frac{d^m\bar{x}}{dt^m}, \bar{x}_D(t)\right] \tag{2.34}$$

$$\bar{v}_{NL}(t) = u\left[\bar{x}(t), \frac{d\bar{x}}{dt}, \cdots, \frac{d^m\bar{x}}{dt^m}, \bar{x}_D(t)\right] \tag{2.35}$$

The currents and voltages present at each of the nonlinear device ports is represented by vectors $\bar{i}_{NL}(t)$ and $\bar{v}_{NL}(t)$ [10]. The vector of state variables is defined as $\bar{x}(t)$ and the vector of time delayed state variables is defined as $\bar{x}_D(t)$. The time-delayed state variable vector $\bar{x}_D(t)$ can be written as a function of time (t) with a time delay τ_i, i.e. $(\bar{x}_D(t))_i = \bar{x}_i(t-\tau_i)$. The time delay term τ_i can be a function of the state variables $\bar{x}(t)$ [10]. The size of the all vectors in equations (2.34) and (2.35) are the same and equal to the number of ports between the sub-circuits [10]. The use of parametric equations allows for complete generality in the modeling of

each nonlinear device.

The use of harmonic balance requires the availability of port voltage and currents in the frequency domain. So far equations have been written and formulated in the time domain. The conversion to the frequency domain is accomplished using a vector of time domain samples of the i^{TH} state variable given by,

$$\bar{\boldsymbol{x}}_i = [x_i(t_0), x_i(t_1), \cdots, x_i(t_{2\mathrm{m}}),]^T \tag{2.36}$$

Converting all of the frequency based state variables to the time domain is accomplished as follows:

$$\bar{\boldsymbol{x}}_i = \mathcal{F}^{-1}(\bar{X}_i) \tag{2.37}$$

For one device having n states, the state-variable vector in frequency domain is defined as follows,

$$\bar{X} = \begin{bmatrix} \bar{X}_1 \\ \bar{X}_2 \\ \vdots \\ \bar{X}_n \end{bmatrix} \tag{2.38}$$

Each state from Equation (2.38) is comprised of individual frequency components (m),

$$\bar{X}_i = \begin{bmatrix} X_{i,0} \\ X_{i,1} \\ \vdots \\ X_{i,m} \end{bmatrix} \tag{2.39}$$

The time-domain vectors of the derivatives and time-delayed state variables are obtained using the following equations:

$$\frac{d\,\bar{\mathbf{x}}_i(t)}{dt} = \mathcal{F}^{-1}(j\,\Omega\,\bar{X}_i) \quad \text{where} \quad \Omega = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \omega_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m\omega_0 \end{bmatrix} \tag{2.40}$$

$$\bar{\mathbf{x}}_{D(i)}(t) = \mathcal{F}^{-1}\left(\begin{bmatrix} e^{j\omega_0(0)\tau_{D(i)}} & 0 & \cdots & 0 \\ 0 & e^{j\omega_0(1)\tau_{D(i)}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{j\omega_0(m)\tau_{D(i)}} \end{bmatrix} \bar{X}_i \right) \tag{2.41}$$

The samples of $\bar{i}_{NL}(t)$ and $\bar{v}_{NL}(t)$ are generated using equations (2.34) and (2.35). Finally the Fourier transform is used to obtain the $\bar{I}_{NL}$ and $\bar{V}_{NL}$ vectors in the frequency domain.

$$\bar{I}_{NL(i)} = \mathcal{F}[\,\bar{i}_{NL(i)}(t)] \tag{2.42}$$

$$\bar{V}_{NL(i)} = \mathcal{F}[\,\bar{v}_{NL(i)}(t)] \tag{2.43}$$

where

$$\bar{V}_{NL} = \begin{bmatrix} \bar{V}_{NL(1)} \\ \bar{V}_{NL(2)} \\ \vdots \\ \bar{V}_{NL(n)} \end{bmatrix} \quad \& \quad \bar{I}_{NL} = \begin{bmatrix} \bar{I}_{NL(1)} \\ \bar{I}_{NL(2)} \\ \vdots \\ \bar{I}_{NL(n)} \end{bmatrix} \tag{2.44}$$

2.5.2 fREEDA Architecture and Addition of New Simulation Technique

The fREEDA simulator is programmed mainly using the C++ language. The circuit description is entered as a netlist file that has a format similar to the open source Simulation Program with Integrated Circuit Emphasis (SPICE)[26]. This allows the circuit to be formulated in a standard way which can be used for various simulation types. Internally the code for each circuit analysis type is defined in a separate class. For a particular analysis type to be used a command line is entered in the netlist file with various user configurable options. The device

models are formulated using state variables and adding a new device is done separately of the main program. The addition of a new circuit analysis type requires minimal changes in the core code and consists mainly in the addition of a new class. The circuit data is available to the new class in an internal representation format (a graph).

In fREEDA the frequency domain vectors $\bar{I}_{NL}$ and $\bar{V}_{NL}$ are directly available as functions of the state variables $\bar{X}$ (also in the frequency domain). This simplifies the task of implementing frequency domain analysis techniques such as harmonic balance.

The programing architecture provides access to the circuit information in various forms such as, modified nodal admittance matrix, source vector, linear network impedance matrix and others. FREEDA uses a circuit partitioning shown in Fig. 2.9 which allows for the easy addition of a new simulation technique.

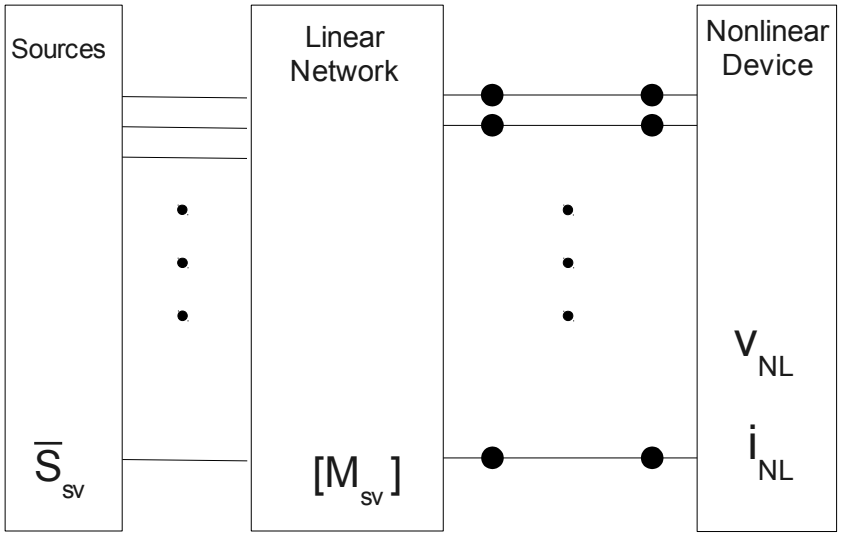

Fig. 2.9 fREEDA Circuit Partition

The selected implementation approach uses the linear network impedance matrix represented by $[M_{SV}]$, containing the linear network information and topology. The $\bar{S}_{SV}$ vector

contains the contribution of the sources to the circuit.

The circuit partitioning required for the wave approach is shown in Fig. 2.10. Each nonlinear device is divided up into its own block with subsequent ports. Each nonlinear device is represented by independent state variables ($\bar{X}$) which are used to calculate the voltages and currents at the ports of nonlinear devices.

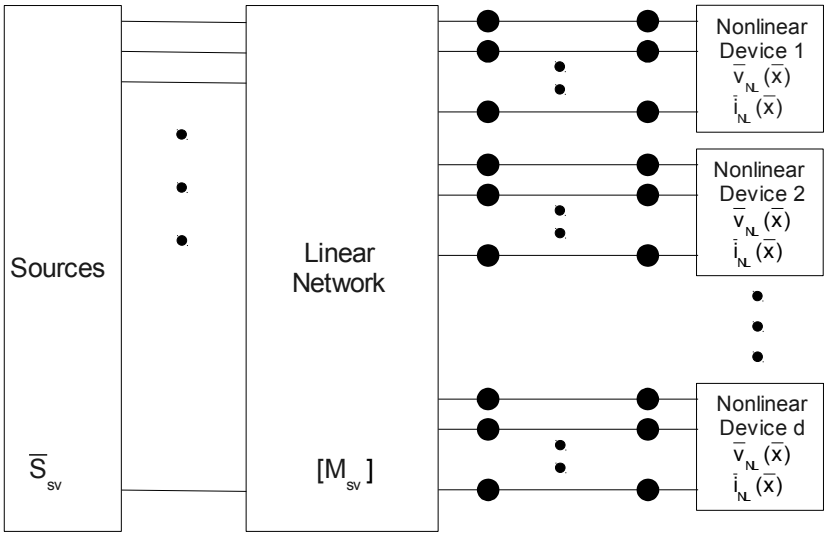

Fig. 2.10 Circuit Partition Used in This Thesis.

The error function within fREEDA is given by,

$$\bar{S}_{sv} - [M_{sv}]\bar{I}_{NL}(\bar{X}) - \bar{V}_{NL}(\bar{X}) = \bar{0} \tag{2.45}$$

The nonlinear devices can have any number of state variables. For example a two terminal device such as a diode has one state variable. A BJT transistor with emitter, base and collector would be a two state variable device.

2.5.3 Example Netlist in fREEDA

An example of a simple netlist used in fREEDA is shown in Fig. 2.11.

```
*** Wave HB Simple Diode Circuit ***

*Simulation options
.options freq=1e8

*Simulation Type and setup parameters
.wavehb n_freqs=15  fundamental=freq tol=1e-8 usecaps=0 n_iter=100  zref=800

*Circuit netlist
vsource:v1 1 0 f=freq vac=5 phase=-90
resistor:r1 1 2 r=1k
diode:d1 2 0 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.

*Simulation Output
.out plot term 2 vf invfft 2 repeat in "simplediodev.whb"
.out plot term 1 vf term 2 vf sub invfft 2 repeat in "simplediodеresv.whb"
.end
```

Fig. 2.11 Example Netlist

The first two lines are comments marked at the beginning by an '*'. The .options line is where various simulation names can be defined in this example the frequency is set to 1e8. The next line after the comment selects the type of simulation to run. In this case the wave harmonic balance (Wave HB) simulation technique is called (.wavehb). The various setup options for the simulation technique are also included on this line. The number of frequencies considered for the harmonic balance simulaiton is 15, the fundamental frequency is 'freq', the solution tolerance is set to 1e-8, the number of Newton iterations for each nonlinear device is 100 and the reference impedance is 800**Ω**. The next few lines represent the circuit configuration and values. The vsource, resistor and diode represent the type of component in the circuit. The following numbers are the nodes the device is connected to and the specific values and parameters of the device. The final lines set the output data to be saved when the simulation is complete. For this simulation the voltage at the specified nodes is converted from the frequency domain (invfft) to produce the voltage with respect to time waveform plot.

Chapter 3

Wave Based Harmonic Balance

3.1 Introduction

This chapter presents a formulation of HB using power waves that expands the techniques discussed in Section 2.4.2, i.e., Kerr's multiple reflections [3] and Hicks and Khan's voltage updates [4]. The use of power waves could improve the convergence of the system because conservation of power restraints can be implemented on the reflections to prevent possible divergence situations; if a device is passive then the power of the reflected waves must be less than or equal to the power of the incident waves. The wave-based HB technique is implemented and demonstrated in fREEDA. This chapter covers the formulation of equations, flow of the algorithm, convergence analysis, convergence improvements, programming and implementation, preconditioner applications and problems encountered.

Section 3.2 presents the formulation of the HB equations, harmonic balance solution process and programing implementation. Section 3.3 analyzes the convergence of the proposed wave HB method and Section 3.4 proposes convergence improvements. Section 3.5 explores the possibilities of the wave HB method being used as a preconditioner. Section 3.6 highlights problems with implementation and their solutions.

3.2 fREEDA Programing and Implementation

As previously discussed fREEDA is used to implement the proposed wave HB technique. Starting from the general form between the linear and nonlinear circuit partition discussed in Section 2.5, we need to interface with fREEDA. For this some transformations of circuit information and programing structure must be performed. This is broken into three parts, equation formulation, solution process, and implementation.

3.2.1 fREEDA Equation Formulation (Linear)

The first step for the proposed method is connecting fictitious transmission lines between the linear and nonlinear networks. The characteristic impedance of the transmission lines is denoted as Z_{ref} and their length is set to zero. The characteristic impedance is the same for all transmission lines connected between the linear and nonlinear ports.

Incident power waves $\bar{A}$, $\bar{A}'$ and reflected power waves $\bar{B}$, $\bar{B}'$ are defined as shown in Fig. 3.1. The direction of the incident and reflected power waves differs from that of a conventional two port network by referencing to the nonlinear devices.

With the use of transmission line theory the network matrix $[M_{SV}]$ must be transformed into a scattering parameter matrix ($[S]$). The transmitted and reflected power waves replace the voltages and currents. The changes to the partitioning and introduction of power waves is shown in Fig. 3.1. In this section the power waves ($\bar{A}$ & $\bar{B}$) and port voltage ($\bar{V}_{NL}$) and currents ($\bar{I}_{NL}$) are valid for all ports at one frequency.

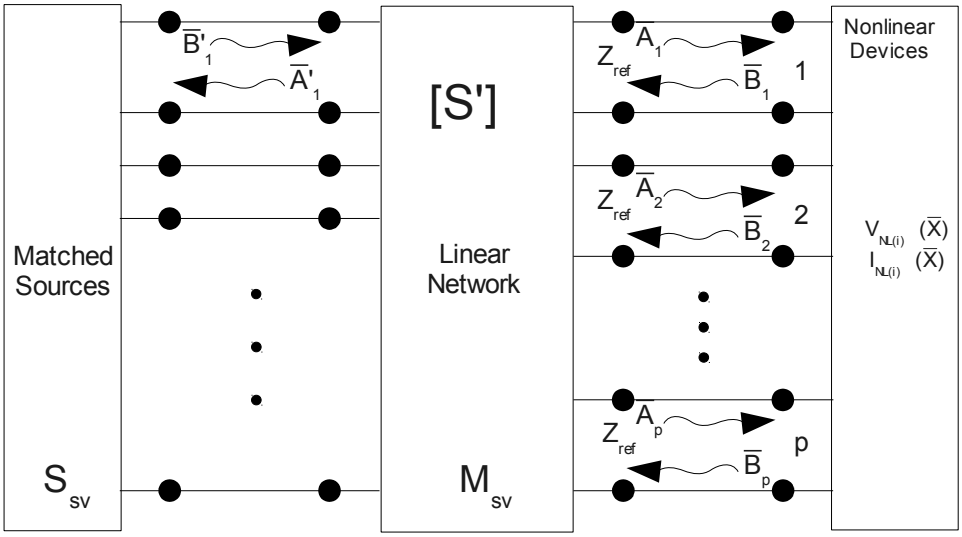

Fig. 3.1 Partition with Fictitious Transmission Lines and Power Waves

For each frequency the scattering parameter matrix ($[S']$) represents a multi-port network with two groups of ports,

$$\begin{bmatrix} \bar{A} \\ \bar{A}' \end{bmatrix} = [S'] \begin{bmatrix} \bar{B} \\ \bar{B}' \end{bmatrix} = \begin{bmatrix} [S_{11}] & [S_{12}] \\ [S_{21}] & [S_{22}] \end{bmatrix} \begin{bmatrix} \bar{B} \\ \bar{B}' \end{bmatrix} \tag{3.1}$$

The network topology and linear device values are contained in the scattering matrix. The individual blocks in equation (3.1) represent the relationships between the incident and reflected waves at the two port groups. Since the sources are assumed to be matched, $\bar{B}'$(vector) is independent of $\bar{A}'$. Thus $\bar{A}'$ is not needed to calculate the waves incident to the nonlinear device ports (p) and $\bar{A}$ is given by:

$$\bar{A} = [S_{11}]\bar{B} + [S_{12}]\bar{B}' \tag{3.2}$$

Defining $\bar{A}_0 = [S_{12}]\bar{B}'$ and $[S] = [S_{11}]$, respectively leads to,

$$\bar{A} = [S]\bar{B} + \bar{A}_0 \ . \tag{3.3}$$

The relation between $[M_{SV}]$, $\bar{S}_{SV}$, $[S]$ and $\bar{A}_0$ will be derived next. Starting from the original error function provided in fREEDA given in equation (2.45) and repeated here for convenience,

$$\bar{S}_{sv} - [M_{sv}]\bar{I}_{NL}(\bar{X}) - \bar{V}_{NL}(\bar{X}) = \bar{0} \quad , \tag{3.4}$$

a relationship between $\bar{V}_{NL}$, $\bar{I}_{NL}$ and $\bar{A}$, $\bar{B}$ is required. Defining a diagonal matrix $[D]$ of size $p \times p$ which contains the square root of the characteristic impedance of the transmission lines and has a structure of,

$$[D] = \begin{bmatrix} \sqrt{Z_{ref}} & 0 & 0 & 0 \\ 0 & \sqrt{Z_{ref}} & 0 & 0 \\ 0 & 0 & \ldots & \ldots \\ 0 & 0 & \ldots & \sqrt{Z_{ref}} \end{bmatrix} \tag{3.5}$$

$\bar{I}_{NL}$ and $\bar{V}_{NL}$ can be written as follows:

$$\bar{V}_{NL} = [D](\bar{A}+\bar{B}) \tag{3.6}$$

$$\bar{I}_{NL} = [D]^{-1}(\bar{A}-\bar{B}) \tag{3.7}$$

Substituting Equation (3.6) and Equation (3.7) into the original fREEDA error function Equation (3.4) the new equation is formed.

$$\bar{S}_{sv} - [M_{sv}][D]^{-1}(\bar{A}-\bar{B}) - [D](\bar{A}+\bar{B}) = \bar{0} \tag{3.8}$$

Rearranging Equation (3.8) to isolate for the incident power wave $\bar{A}$ yields,

$$\bar{A} = ([M_{sv}][D]^{-1}+[D])^{-1}[-\bar{S}_{sv} + ([M_{sv}][D]^{-1}-[D])\bar{B}] \tag{3.9}$$

Multiplying $([M_{sv}][D]^{-1}+[D])^{-1}$ into $[-\bar{S}_{sv} + ([M_{sv}][D]^{-1}-[D])\bar{B}]$ the Equation (3.9) is arranged to match the structure of Equation (3.3).

$$\bar{A}_m = ([M_{sv}][D]^{-1}+[D])^{-1}([M_{sv}][D]^{-1}-[D])\bar{B}_m + ([M_{sv}][D]^{-1}+[D])^{-1}\bar{S}_{sv} \quad (3.10)$$

From this a new definition of $[S]$ and $\bar{A}_0$ is obtained:

$$[S] = ([M_{sv}][D]^{-1}+[D])^{-1}([M_{sv}][D]^{-1}-[D]) \quad (3.11)$$

$$\bar{A}_0 = ([M_{sv}][D]^{-1}+[D])^{-1}\bar{S}_{sv} \quad (3.12)$$

With a definition of $[S]$ and $\bar{A}_0$ their values are calculated. The parameters of the scattering matrix and source vector change with respect to frequency. This requires independent $[S_m]$ and $\bar{B}_{0,m}$ terms for each harmonic (m) plus DC considered. Each element of the scattering matrix and source vector are also comprised of complex values.

3.2.2 fREEDA Equation Formulation (Nonlinear)

In this section the power waves ($\bar{A}$ & $\bar{B}$), state variables ($\bar{X}$) , port voltages ($\bar{V}_{NL}$) and port currents ($\bar{I}_{NL}$) are valid for all harmonics (m) plus DC for one nonlinear device (d). The vector containing power wave reflections from the nonlinear devices ($\bar{B}_m$) is calculated using Newton's method. Combining Newton's technique with the use of state variables requires the development of both the update equations and the error functions for each port. The error functions are based from Equation (3.6) and Equation (3.7) by moving all terms to the right and defining error functions $\bar{f}_v$ and $\bar{f}_i$ as follows:

$$\bar{f}_v = D^{-1}\bar{V}_{NL} - \bar{A} - \bar{B} \quad (3.13)$$

$$\bar{f}_i = D\bar{I}_{NL} - \bar{A} + \bar{B} \quad (3.14)$$

There is a set of voltage and current error functions defined for each nonlinear device and each frequency and the $[D]$ matrix has dimensions $(n\times(m+1))\times(n\times(m+1))$ where,

($n = number\ of\ states$ and $m = number\ of\ harmonics$). The $[D]$ matrix is comprised of diagonal elements of $\sqrt{Z_{ref}}$ the same as in Section 3.2.1.

With the use of Newton's method (Section 2.2.1) the formulation of the Newton iteration equations is as follows,

$$\begin{bmatrix} \frac{\partial f_v}{\partial X} & \frac{\partial f_v}{\partial B} \\ \frac{\partial f_i}{\partial X} & \frac{\partial f_i}{\partial B} \end{bmatrix} \begin{bmatrix} X^{k+1} - X^k \\ B^{k+1} - B^k \end{bmatrix} = -\begin{bmatrix} f_v(X^k, B^k) \\ f_i(X^k, B^k) \end{bmatrix}$$

$$\rightarrow \quad \begin{bmatrix} \frac{\partial f_v}{\partial X} & \frac{\partial f_v}{\partial B} \\ \frac{\partial f_i}{\partial X} & \frac{\partial f_i}{\partial B} \end{bmatrix} \begin{bmatrix} X^{k+1} - X^k \\ B^{k+1} - B^k \end{bmatrix} + \begin{bmatrix} f_v(X^k, B^k) \\ f_i(X^k, B^k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.15}$$

Taking the partial derivative of the error function given in Equation (3.13),

$$\frac{\partial f_v}{\partial X} = [D]^{-1} \frac{\partial V_{NL}(X)}{\partial X} = [D]^{-1}[J_v] \quad \text{and} \quad \frac{\partial f_v}{\partial B} = -[I] \tag{3.16}$$

Similarly for Equation (3.14),

$$\frac{\partial f_i}{\partial X} = [D]^{-1} \frac{\partial I_{NL}(X)}{\partial X} = [D]^{-1}[J_i] \quad \text{and} \quad \frac{\partial f_i}{\partial B} = [I] \tag{3.17}$$

Substituting Equation (3.16) and Equation (3.17) into Equation (3.15),

$$\begin{bmatrix} [D]^{-1} J_v & [I] \\ [D] J_i & [I] \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta B \end{bmatrix} + \begin{bmatrix} f_v \\ f_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.18}$$

And solving for ΔB and ΔX leads to,

$$\Delta \bar{B} = -\bar{f}_i - [D][J_i] \Delta \bar{X} \tag{3.19}$$

and

$$\Delta \bar{X} = ([D][J_i] + [D]^{-1}[J_v])^{-1}(-\bar{f}_i - \bar{f}_v) \tag{3.20}$$

Defining the updates for one nonlinear device ($\bar{X}_d$ and $\bar{B}_d$) from (3.19) and (3.20) is given by,

$$\Delta \bar{X}_d = ([D][J_{i(d)}] + [D]^{-1}[J_{v(d)}])^{-1}(-\bar{f}_{i(d)} - \bar{f}_{v(d)}) \tag{3.21}$$

$$\Delta \bar{B}_d = -\bar{f}_{i(d)} - [D][J_{i(d)}]\Delta \bar{X}_d \tag{3.22}$$

Where $[J_{v(d)}]$ and $[J_{i(d)}]$ are the Jacobian matrices of the port voltage and currents for the nonlinear device being calculated using the state variables from Section 2.5.1.

Harmonic balance technique requires the state variables in the frequency domain. fREEDA provides the Jacobian matrix, state variables, port voltages and currents directly, using the process discussed in section 2.5.1. Combining the linear and nonlinear calculations together produces a fixed point iterations technique given by,

$$\bar{A}_m^{k+1} = [S]_m \bar{B}_m^k + \bar{A}_{0(m)} \tag{3.23}$$

Each nonlinear device's reflected power wave ($\bar{B}_d$) is calculated independently for each nonlinear device considering all frequencies simultaneously. When all of the nonlinear device's reflected waves have been solved for the current iteration (k), Equation (3.23) is independently calculated for each harmonic considered using the reflected waves provided from the nonlinear device's Newton iterations. The solution is found when Equation (3.23) is satisfied for all harmonics. A more detailed explanation of this process is discussed in the next in Section.

3.2.3 Solution Methodology

The iterative process to solve for the solution is now discussed. The solution process

consists of solving for the transmitted and reflected waves. This is accomplished by solving for the local reflection $\bar{B}$ given the transmitted wave $\bar{A}$ for each port(s) of the nonlinear devices. The process uses fixed point iterations where the new transmitted wave is calculated using the previously calculated reflection using Equation (3.23).

The value of the source vector is available from the simulation setup and is therefore used as an initial guess for the transmitted power wave $A = A_0$. The state variables (X) are set to zero as the initial guess and are used to calculate $\bar{I}_{NL}(\bar{X})$ and $\bar{V}_{NL}(\bar{X})$ from the device models within fREEDA. The results of the $\bar{I}_{NL}(\bar{X})$ and $\bar{V}_{NL}(\bar{X})$ calculation is then used in the error functions Equation (3.13) and Equation (3.14).

If the errors $\bar{f}_{i(d)}$ and $\bar{f}_{v(d)}$ are less than the solution tolerance then the correct reflection value $\bar{B}_d$ for the transmitted wave $\bar{A}_d$ has been solved for that device's port(s). If the calculated reflection does not satisfy Equation (3.13) and Equation (3.14) then the state variables and reflected wave are updated by adding $\Delta \bar{X}_d$ and $\Delta \bar{B}_d$ calculated from Equation (3.21) and Equation (3.22) to $\bar{X}_d$ and $\bar{B}_d$. The use of a scaling factor is also used to limit the amount of change from one iteration to the next. This scaling factor α is set to a value between $0 < \alpha < 1$ and the update is calculated by:

$$\bar{X}_d^{k+1} = \bar{X}_d^k + \alpha \Delta \bar{X}_d \tag{3.24}$$

$$\bar{B}_d^{k+1} = \bar{B}_d^k + \alpha \Delta \bar{B}_d \tag{3.25}$$

Once all reflected waves $\bar{B}_d$ have been solved for each port, they are substituted into Equation (3.23) to calculate the new value of $\bar{A}_d$. The iteration process is shown in Fig. 3.2.

This process continues until the state variables are found which satisfy Equation (3.23).

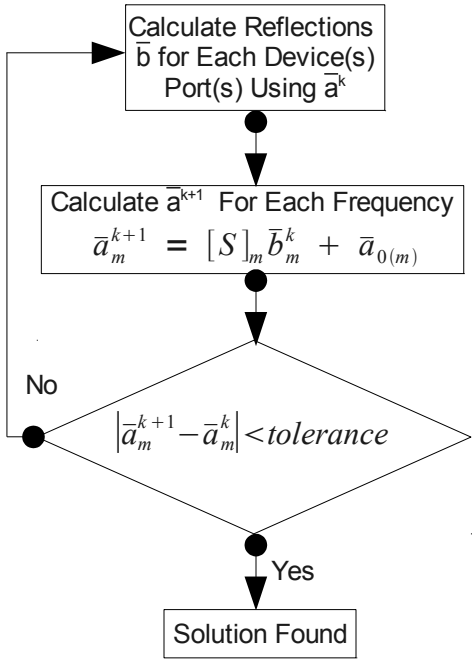

Fig. 3.2 Solution Iteration Process

With the solution process explained the next step is implementing it in software. This becomes a large task having to support circuits of varying size and harmonics. Most of the variables and terms from above are multidimensional matrices or vectors. Certain parts of the solution process are separated for each frequency and others require all frequencies together. This requires care and proper planning to successfully achieve an efficient program. The programming process is discussed in the next section.

3.2.4 Implementation in fREEDA

As mentioned in Section 2.5 the platform selected to integrate the wave technique into was fREEDA. The use of fREEDA greatly reduced the programing effort while increasing the flexibility. The addition of a new class 'wavehb' was added to the simulation package to support wave based harmonic balance. The existing simulation class 'SVHB' or state variable harmonic

balance class was used as an intial model to implement the new class.

Besides the major changes to the equation formulation discussed in Section 3.2.1 and 3.2.2 the use of an open source matrix library was added. The Matrix Template Library (MTL) [22] was selected to ease programing complexity and decrease programing time. The MTL includes various matrix and vector operations for both real and complex numbers. The use of this library did however have some limitations and problems, which are discussed in Section 3.6. The original SVHB simulation was formulated using different vector representations and a different complex number format. Conversions were required to fit the data into a usable format for using the MTL.

With the data rearranged to fit the MTL format the scattering matrices $[S_m]$ and source vectors $\bar{A}_{0(m)}$ are calculated for each frequency using the formulas derived in Section 3.2.1. There is one scattering matrix and source vector per frequency. The scattering parameters are stored in a square matrix with a dimension equal to the number of ports p . An array of scattering matrices is formed to store the values for each frequency. The size of the array is equal to the number of harmonics considered plus DC shown in Equation (3.26).

$$[S_m] = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1p} \\ s_{21} & s_{22} & \cdots & s_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1} & s_{p2} & \cdots & s_{pp} \end{bmatrix} \tag{3.26}$$

Similarly an array is defined for the source vectors. The source vector length is equal to the number of ports and the number of array elements is equal to the number of harmonics plus DC. Both the scattering matrices and source vectors are only calculated once at the beginning of the simulation. This is beneficial because it keeps the computationally expensive matrix

decompositions to a minimum. The structure of the A vector is shown here:

$$\bar{A}_{0(m)} = \begin{bmatrix} A_{0(0)} \\ A_{0(1)} \\ \vdots \\ A_{0(p)} \end{bmatrix} \tag{3.27}$$

The other main values that need to be stored are the vectors $\bar{B}$, $\bar{I}_{NL}(\bar{X})$ and $\bar{V}_{NL}(\bar{X})$. The treatment of $\bar{I}_{NL}(\bar{X})$ and $\bar{V}_{NL}(\bar{X})$ is similar to that of the source vectors by using an array. The array length for $\bar{I}_{NL}(\bar{X})$ and $\bar{V}_{NL}(\bar{X})$ is equal to the number of ports not necessarily equal to the number of devices. For devices that have more than one state variable the dimension of the vector increases to the number of frequencies times the number of states variable for that device shown in Equation (3.28). This requirement adds some programing complexity because many circuits can have a mixture of devices with various states per device. The structure for the vectors in one device is illustrated below:

$$\bar{V}_{NL(d)} = \begin{bmatrix} V_{NL(1,0)} \\ V_{NL(1,1)} \\ \vdots \\ V_{NL(1,m)} \\ V_{NL(2,0)} \\ V_{NL(2,1)} \\ \vdots \\ V_{NL(2,m)} \\ V_{NL(p,0)} \\ V_{NL(p,1)} \\ \vdots \\ V_{NL(p,m)} \end{bmatrix} \quad \bar{I}_{NL(d)} = \begin{bmatrix} I_{NL(1,0)} \\ I_{NL(1,1)} \\ \vdots \\ I_{NL(1,m)} \\ I_{NL(2,0)} \\ I_{NL(2,1)} \\ \vdots \\ I_{NL(2,m)} \\ I_{NL(p,0)} \\ I_{NL(p,1)} \\ \vdots \\ I_{NL(p,m)} \end{bmatrix} \tag{3.28}$$

Unlike the $\bar{A}_0$, $\bar{I}_{NL}(\bar{x})$ and $\bar{V}_{NL}(\bar{x})$ vectors which are stored in an array of vectors the

power wave vectors $\bar{A}$ and $\bar{B}$ are stored as a matrix shown in Equation (3.29). The matrix dimensions being the number of ports by the number of frequencies. This convention allows for easy access to different data. For example the magnitude and phase of all the power waves at one port can be used by referencing an individual row. Similarly referencing one column allows access to all components of one particular frequency.

$$\bar{A} = \begin{bmatrix} a_{1,0} & a_{1,1} & \cdots & a_{1,m} \\ a_{2,0} & a_{2,1} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,0} & a_{p,1} & \cdots & a_{p,m} \end{bmatrix} \qquad \bar{B} = \begin{bmatrix} b_{1,0} & b_{1,1} & \cdots & b_{1,m} \\ b_{2,0} & b_{2,1} & \cdots & b_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p,0} & b_{p,1} & \cdots & b_{p,m} \end{bmatrix} \tag{3.29}$$

The iteration process is set to sequentially loop through the devices and solve the local reflections $\bar{B}$. This is accomplished by formulating the voltage and current Jacobian matrix for the device in question. Both Jacobian matrices $[J_i]$ and $[J_v]$ are square and of the same dimension. Their dimension is equal to the number of ports (p) for the device times the number of frequencies considered. The structure of the current Jacobian matrix $[J_i]$ for one device (d) is,

$$[J_{i(d)}] = \begin{bmatrix} [J_{1,1}] & [J_{1,2}] & \cdots & [J_{1,p}] \\ [J_{2,1}] & [J_{2,2}] & \cdots & [J_{2,p}] \\ \vdots & \vdots & \ddots & \vdots \\ [J_{p,1}] & [J_{1,2}] & \cdots & [J_{p,p}] \end{bmatrix} \tag{3.30}$$

Each term in Equation (3.30) is comprised of another sub-matrices $[J_{r,q}]$ which refers to the number of ports per device with $1 < r < p$ and $1 < q < p$. Each of the sub-matrices $[J_{r,q}]$ are also square and have a dimension of the number of frequencies.

$$[J_{r,q}] = \begin{bmatrix} j_{0,0} & j_{0,2} & \cdots & j_{0,m} \\ j_{1,0} & j_{1,2} & \cdots & j_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ j_{m,1} & j_{m,2} & \cdots & j_{m,m} \end{bmatrix} \tag{3.31}$$

Each element in $[J_{r,q}]$ is composed of one complex number. The voltage Jacobian is of the same form and is not shown.

The process of calculating the Jacobian for each nonlinear device is not a trivial task and interfacing with fREEDA requires great care as the data types and pointers are not directly compatible. The calculation process of the Jacobian matrix is performed using the same structure and process used for converting the state variable based current and voltage vectors from the time to frequency domain as discussed in Section 2.5.1.

The Jacobian is used to calculate the updates to $\bar{X}$ and $\bar{B}$ .This process requires one matrix decomposition for the $([D][J_i] + [D]^{-1}[J_v])^{-1}$ calculation in Equation (3.20). There is a scaling factor (α) applied to the update values for $\Delta \bar{X}$ and $\Delta \bar{B}$ (Section 3.2.3) . This scaling factor is used to help convergence. For each device the stop criteria is met when the infinity norm of $\sqrt{Z_{ref}}(\Delta \bar{A} + \Delta \bar{B})$, $\sqrt{Z_{ref}}\,\bar{f}_i$, $\bar{f}_v / \sqrt{Z_{ref}}$, $\Delta \bar{X}$ are all less then the specified simulation tolerance for that port.

Implementation of the overall programing including debugging and verification added up to approximately thirteen hundred lines of code. This does not include the trials with alternative nonlinear equation solvers and the code used to integrate with Octave files [25] to verify parts of the code that were operational during development.

3.3 Convergence Analysis

3.3.1 Power Bound

The use of a relaxation technique based on power waves for harmonic balance analysis is guaranteed not to diverge to infinity, even if the initial guess is far from the solution. This property is a direct consequence of using power waves. This is important because it ensures that there is a physically meaningful excitation for the device models therefore avoiding numerical problems. The fixed point relaxation scheme can be summarized using the following equation,

$$\bar{\boldsymbol{B}} = \bar{F}([\boldsymbol{S}]\bar{\boldsymbol{B}} + \bar{\boldsymbol{A}}_{\boldsymbol{0}}) \tag{3.32}$$

where the bold matrix ($[\boldsymbol{S}]$) and vectors ($\bar{\boldsymbol{B}}$, $\bar{\boldsymbol{A}}_0$)are defined as,

$$[\boldsymbol{S}] = \begin{bmatrix} [S_0] & 0 & \cdots & 0 \\ 0 & [S_1] & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & [S_m] \end{bmatrix} \tag{3.33}$$

$$\boldsymbol{B}^{k+1} = \begin{bmatrix} B_0^{k+1} \\ B_1^{k+1} \\ \vdots \\ B_m^{k+1} \end{bmatrix} \tag{3.34}$$

$$\bar{\boldsymbol{A}}_0 = \begin{bmatrix} A_{0(0)} \\ A_{0(1)} \\ \vdots \\ A_{0(m)} \end{bmatrix} \tag{3.35}$$

The total power reflected at each iteration of Equations (3.23) is given by $|\bar{\boldsymbol{B}}|^2$, where the bars denote the 2-norm of the vector. Since the nonlinear devices are assumed to be passive,

$$|\bar{\boldsymbol{B}}^{k+1}|^2 = |F([\boldsymbol{S}]\bar{\boldsymbol{B}}^{k}+\bar{\boldsymbol{A}}_0)|^2 \leqslant \left|L_{Sc}([\boldsymbol{S}]\bar{\boldsymbol{B}}^{k}+\bar{\boldsymbol{A}}_0)\right|^2 , \tag{3.36}$$

where L_{Sc} is a scalar with a value between $0 \leqslant L_{Sc} < 1$. If it is assumed that the upper bound is propagated during the first iteration the result obtained is as follows:

$$|L_{Sc}([\boldsymbol{S}]\bar{\boldsymbol{B}}^{k}+\bar{\boldsymbol{A}}_0)| \leqslant L_{Sc}(|[\boldsymbol{S}]\bar{\boldsymbol{B}}^{k}|+|\bar{\boldsymbol{A}}_0|) \leqslant L_{Sc}(|\bar{\boldsymbol{B}}^{k}|+|\bar{\boldsymbol{A}}_0|) \tag{3.37}$$

Given $\bar{\boldsymbol{B}}_0$,

$$\begin{aligned} \bar{\boldsymbol{B}}^{k+1} &\leqslant L_{Sc}(|\bar{\boldsymbol{B}}^{k}|+|\bar{\boldsymbol{A}}_0|) \\ &\vdots \\ \bar{\boldsymbol{B}}^{k+1} &\leqslant L_{Sc}^{k+1}(|\bar{\boldsymbol{B}}^{k}|+|\bar{\boldsymbol{A}}_0|)+(L_{Sc}^{k}+L_{Sc}^{k-1}+\cdots+L_{Sc})|\bar{\boldsymbol{A}}_0| \end{aligned} \tag{3.38}$$

Therefore,

$$\lim_{k\to\infty} |\bar{\boldsymbol{B}}^{k+1}|^2 \leqslant \frac{1}{1-L_{Sc}}(L_{Sc}|\bar{\boldsymbol{A}}_0|^2) \tag{3.39}$$

3.4 Convergence Improvements

The proposed method using power waves demonstrates good empirical convergence properties with simple circuits however there were some convergence issues when simulating relatively complex and highly nonlinear circuits. To help overcome some of the convergence issues a few techniques detailed below have been developed in this Thesis.

3.4.1 Addition of Parallel Capacitors

The capacitors are connected to each port as seen in Fig. 3.3. These capacitors are assumed to be active only in an independent time dimension. A transient analysis is performed in this time dimension following a procedure inspired by the pseudo transient analysis technique presented in [27] and the multi-time techniques presented in [21]. The addition of port capacitors after the discretization in time domain can be seen equivalent to adding a conductance in parallel with the port. This conductance can be adjusted based on the size of the capacitance or the time step in the pseudo-transient analysis.

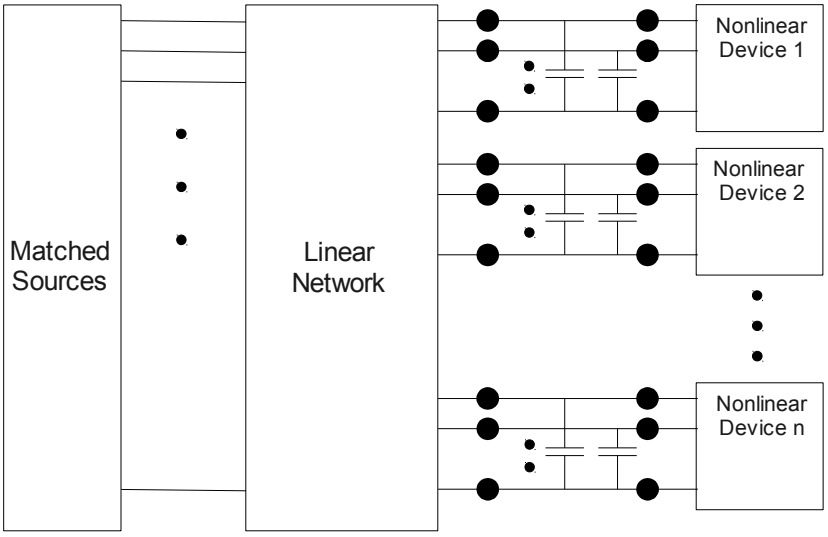

Fig. 3.3 Addition of Parallel Port Capacitors

Implementing this technique into the wave based harmonic balance approach is accomplished by transforming the capacitors into an equivalent circuit. The current in a capacitor is related to the capacitance and the change of voltage with respect to the independent time variable (t'),

$$I_c \simeq C \frac{dV}{dt'} \tag{3.40}$$

The derivative in equation (3.41) can be approximated with the Backward Euler formula by using a small time step h ,

$$I_c \simeq C \frac{V(t'+h)-V(t')}{h} \tag{3.41}$$

Rearranging Equation (3.41) into Equation (3.42) and defining a current term I and a conductance term G_1 the equivalent circuit can be formed (Fig. 3.4).

$$I_c \simeq \frac{-CV(t')}{h} + \frac{CV(t'+h)}{h} \tag{3.42}$$

$$I = \frac{-CV(t')}{h} \qquad G_1 = \frac{1}{R} = \frac{C}{h} \tag{3.43}$$

$$I_c \simeq I(V^u) + G_1(V^{u+1}) \tag{3.44}$$

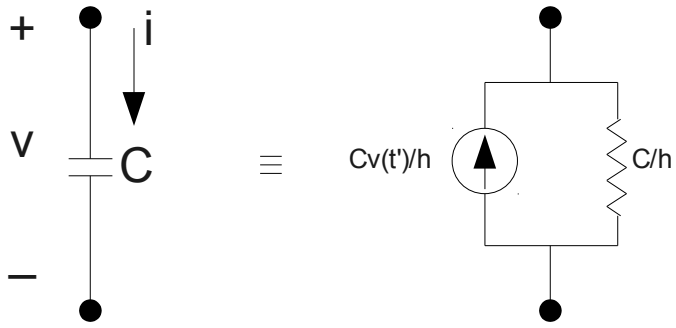

Fig. 3.4 Capacitor Transformed into Equivalent Circuit

Replacing the capacitors with its equivalent circuit (Fig. 3.4) requires some slight modifications to the equations derived in Section 3.2. The first equation to be modified is the nonlinear current returned from the device models.

$$\bar{i}_{NL}(\bar{x}) \rightarrow \bar{i}_{NL}(\bar{x}) + [G](\bar{v}_{NL}^{k}(\bar{x}, t+h) - \bar{v}_{NL}^{k-1}(\bar{x}, t)) \tag{3.45}$$

Where the square diagonal matrix is defined as,

$$[G] = \begin{bmatrix} G_1 & 0 & \cdots & 0 \\ 0 & G_1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & G_1 \end{bmatrix} \tag{3.46}$$

and the dimensions of $[G]$ are equal to the number of state variables times the number of frequencies considered for the device being calculated.

For the currents to remain balanced the effect of the capacitors must be considered. As can be seen in Equation (3.45) as the port voltage approaches steady state the current related to the capacitors goes to zero.

Similar modifications are made to the nonlinear current error function in Equation (3.14). The additional current is added to the port nonlinear current in the error function shown in Equation (3.47).

$$\bar{f}_i \rightarrow \bar{f}_i = [D][\bar{i}_{NL}(\bar{x}) + [G](\bar{v}_{NL}^{k}(\bar{x}, t+h) - \bar{v}_{NL}^{k-1}(\bar{x}, t))] - \bar{a} + \bar{b} \tag{3.47}$$

The final affected equation is the current Jacobian matrix. Again the current caused by the capacitors must be accounted for as seen in.

$$[J_i] \rightarrow [J_i] + [G][J_v] \tag{3.48}$$

The value of $[G]$ is a main diagonal matrix therefore it can be used to change the properties of the Jacobian matrix.

As the pseudo transient reaches steady state the capacitors effect on the solution are automatically removed without changing any simulation parameters. The value of conductance is used to passivate the port so at the onset of the iteration process the impedance seen at the port is dominated by the capacitor. As the voltage at the port begins to reach a steady state the effect of the capacitor is negligible and the solution is found.

For simulations using capacitors there are two different analysis available, referred to as timing analysis and iterative timing analysis [23] . Timing analysis requires only one iteration of the main reflection loop Equation (3.23) per time step of the 'pseudo transient' approach. The use of iterative time analysis allows Equation (3.23) to converge to solution before a 'pseudo transient' time step is calculated. Solving the reflections to the solution tolerance when using 'pseudo transient' could have little benefit because the solution calculated is not correct until the port voltage has reach steady state. A flow diagram of the solution process using 'pseudo transient' is shown in Fig. 3.5.

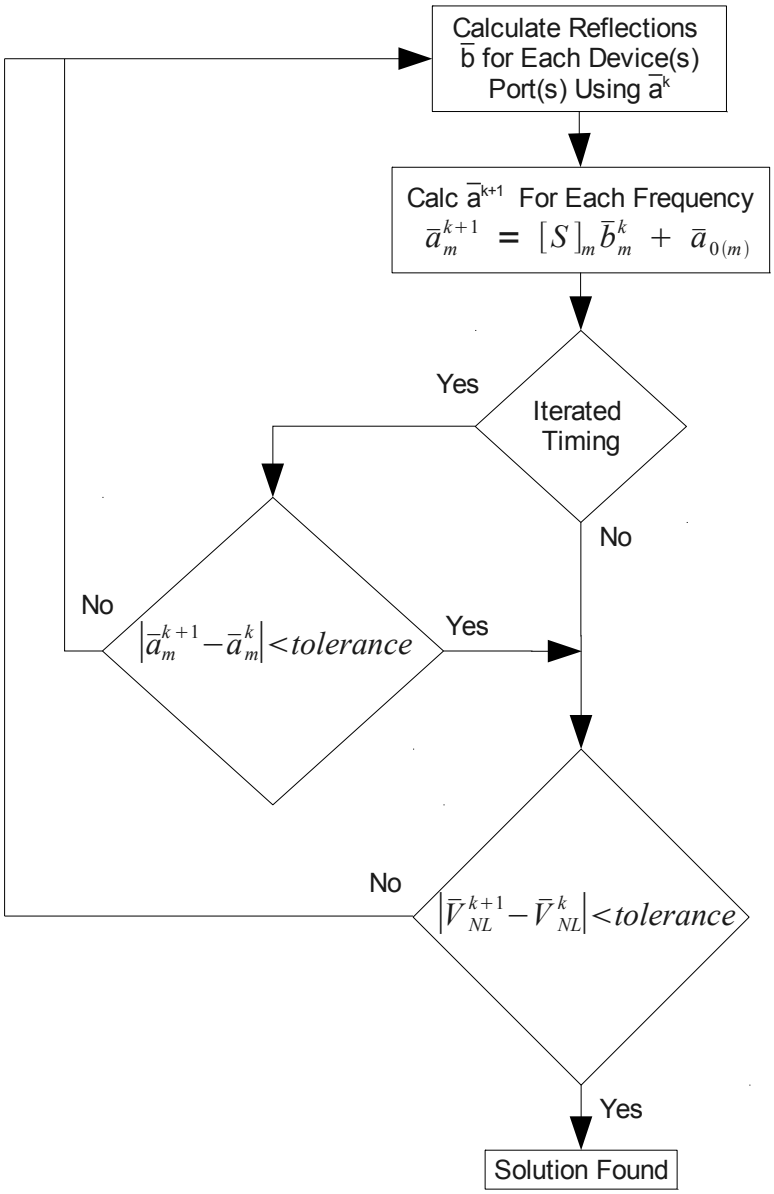

Fig. 3.5 Pseudo Transient Flow Solution Flow Diagram

3.4.2 Extrapolation Techniques

The use of capacitors helps the convergence of the system but as a trade off it extends the solution time considerably under some circumstances. The increase in solution time can be attributed to the slowly changing port voltage especially when close to the steady state value. To overcome this newly introduced problem an extrapolation technique is proposed. The theory for

using extrapolation is that capacitors charge quite quickly when voltage is first applied. However as the port voltage approaches steady state the capacitors take much longer to achieve full charge. So the use of extrapolation in theory should help overcome this slow final charging.

The use of extrapolation is implemented on both the port voltage variable $\bar{v}_{NL}$ and the reflected wave $\bar{B}$. The ability to enable and disable the use of extrapolation for both variables is added to the program through the use of simulation options within the netlist file. The sequence for $\bar{v}_{NL}$ and $\bar{B}$ is generated in the form of $\bar{v}_{NL}^{1}, \bar{v}_{NL}^{2}, \cdots, \bar{v}_{NL}^{k+1}$ and $\bar{B}^{1,} \bar{B}^{2,} \cdots, \bar{B}^{k+1}$ with the number of samples k. The sequence is then used to calculate $\bar{u}_j$ Equation (2.6), $[U]$ Equation (2.7), $\bar{c}$ Equation (2.8) and the final extrapolated solution $\bar{s}$ Equation (2.10) as discussed in Section 2.2.2.

3.4.3 Gradient Descent Method

Another method attempted to improve convergence was that of gradient descent. Gradient descent is used to scale iterations updates in the correct direction. The hope being that if the current iteration is close the gradient descent can be used to guide the updates in the correct direction.

The system of equations in question are currently in the form of a vector. The gradient descent method is performed on a scalar. Given the fixed point relaxation scheme in Equation (3.32) an error function $\bar{G}(\bar{\boldsymbol{B}})$ can be defined as,

$$\bar{G}(\bar{\boldsymbol{B}}) = \bar{F}([\boldsymbol{S}]\bar{\boldsymbol{B}} + \bar{\boldsymbol{A}}_{\boldsymbol{0}}) - \bar{\boldsymbol{B}} = \Delta\bar{\boldsymbol{B}} \tag{3.49}$$

The bold vectors and matrices are defined in Section 3.3.1. The Jacobian of $\bar{F}(\bar{\boldsymbol{B}})$ and the error function ($\bar{G}(\bar{\boldsymbol{B}})$)are defined as $[J_F]$ and $[J_G]$ respectively, where,

$$[J_G] = [J_F][\boldsymbol{S}]-[I] \tag{3.50}$$

Converting the vector error function into a scalar function is accomplished as follows:

$$H(\bar{\boldsymbol{B}}) = \bar{G}^*(\bar{\boldsymbol{B}})\bar{G}(\bar{\boldsymbol{B}}) \quad , \tag{3.51}$$

where $\bar{G}^*(\bar{\boldsymbol{B}})$ is the complex conjugate transpose.

The gradient of $G(\bar{\boldsymbol{B}})$ is then,

$$\begin{aligned} \nabla \bar{H} &= 2[J_G]^* \bar{G}(\bar{\boldsymbol{B}}) \\ &= 2[[\boldsymbol{S}]^*[J_B]^*-[I]]\Delta \bar{\boldsymbol{B}} \\ &= 2[[\boldsymbol{S}]^*[J_B]^* \Delta \bar{\boldsymbol{B}}-\Delta \bar{\boldsymbol{B}}] \quad , \end{aligned} \tag{3.52}$$

where $[J_F]^*$ and $[S]^*$ are the complex conjugate transpose of $[J_F]$ and $[S]$, respectively.

The value of $[J_F]^* \Delta \bar{\boldsymbol{B}}$ is calculated using,

$$[J_F]^* \Delta \bar{\boldsymbol{B}} = [[D^{-1}]-[D][Y]^*][[D^{-1}]+[D][Y]^*]^{-1} \Delta \bar{\boldsymbol{B}} \quad , \tag{3.53}$$

where $[Y]$ and $\Delta \bar{\boldsymbol{B}}$ are defined as,

$$[Y]_B = [J_i][J_v]^{-1} \qquad \Delta \bar{\boldsymbol{B}} = \bar{\boldsymbol{B}}^{k+1} - \bar{\boldsymbol{B}}^k \tag{3.54}$$

One $[Y]_B$ term is calculated for each nonlinear device.

The direction of ∇H is used to determine the update direction. The update ($\Delta \bar{\boldsymbol{B}}$) for the next iteration is normally calculated using Equation (3.21) and Equation (3.22). With the use of gradient descent, the update is scaled using the direction of the gradient field. If the gradient field and the calculated update $\Delta \bar{\boldsymbol{B}}$ are in same direction the update $\Delta \bar{\boldsymbol{B}}$ is flipped to the opposite direction before being added to the current value of $\bar{\boldsymbol{B}}$.

3.4.4 Tolerance Stepping

Tolerance stepping is used to solve the individual nonlinear device reflections $\bar{B}$ to a looser tolerance than the one set in the simulation netlist. This reduces the number of Newton iterations required for each main routine iteration. As the iteration process becomes closer to solution the tolerance of the individual nonlinear device reflection calculations is stepped by a factor of 0.1 until the required simulation tolerance is reached. This process is demonstrated in Fig. 3.6.

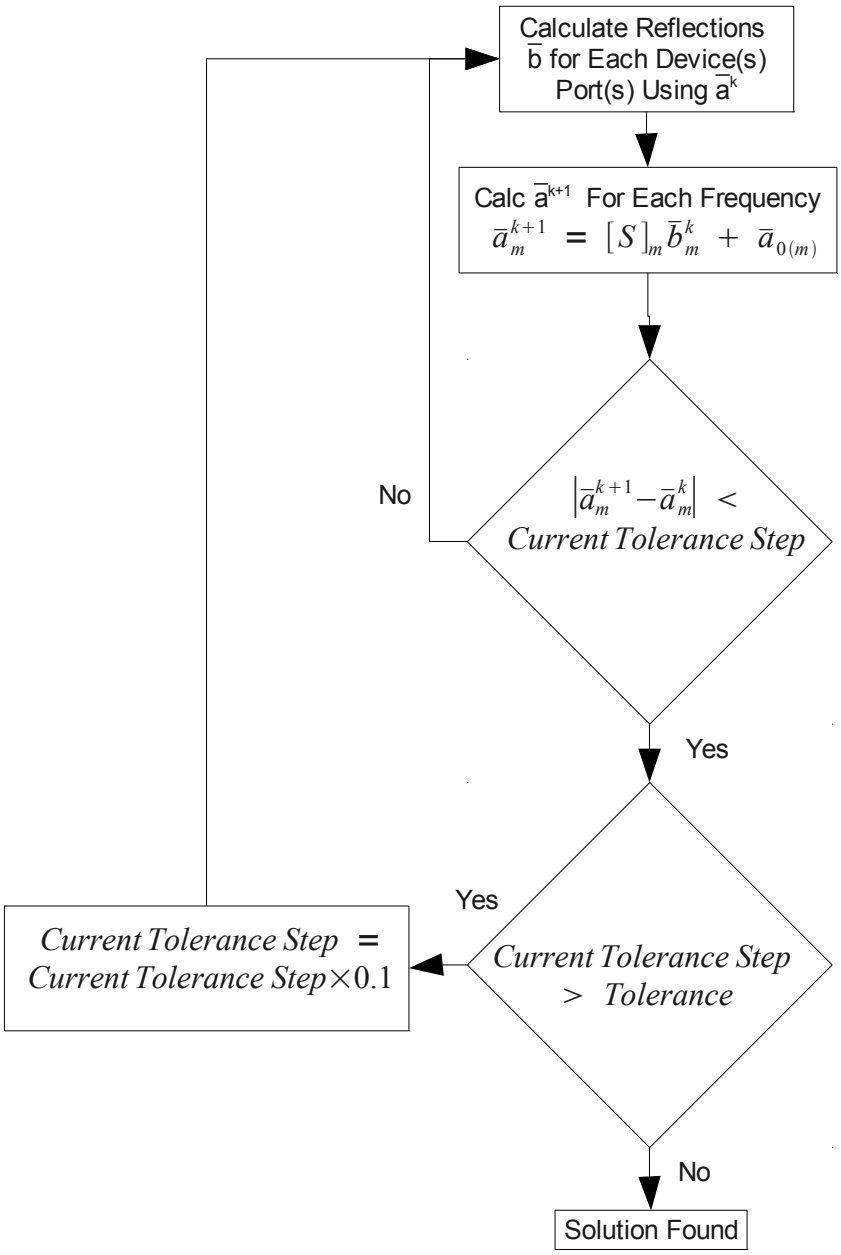

Fig. 3.6 Tolerance Stepping Flow Diagram

3.5 Wave HB as Preconditioner

The wave technique can also be used as a type of preconditioner for the more traditional Newton-based HB simulation. The idea is to allow the wave approach to solve the circuit operation to a somewhat looser tolerance. The results of the loose simulation tolerance is then used as the initial guess for the start of Newton-based HB. The relaxation approach initially convergences quickly and therefore requires less computational time to solve to a slack tolerance. Having the Newton-based HB approach start from an approximate solution should reduce the amount of large Jacobian decompositions and therefore the solution time.

For Newton-based HB the major time dominating process for large circuits is decomposing the Jacobian matrix. The dominating process for the wave HB approach is the accuracy of the solution. The tighter the tolerance the longer the wave HB process takes to converge to a solution. Exploiting the fact that wave HB can solve to a looser tolerance quicker then Newton-based HB a mixture of the two should be able to produce an overall improvement.

3.6 Problems and Solutions

Upon first implementing the wave technique a few simple circuits solved successfully and the solution time was comparable to conventional techniques. Attempting circuits of greater size and complexity, the solution convergence became a serious issue. Various modifications and techniques were attempted to help improve convergence. After many hours of failed attempts further investigation started to point to a serious problem in the MTL.

The matrix template library which is used to store and perform various complex matrix calculations appeared to be incorrectly calculating the LU decomposition. Further investigation

revealed that the LU function did not properly support complex matrices. The solution to this was to use the MTL interface to linear algebra package (LAPACK) routine [24]. The LAPACK routines are a Fortran based matrix library which does support complex matrix decomposition. The matrix template library also lacked a routine to properly perform a matrix complex conjugate transpose. A function was written separately to perform this operation.

To help verify all further modifications and calculations separate routines were written to output matrices and vectors to files. The files are stored in a format that can be read into Octave [25] where calculations can be confirmed to be correct.

Chapter 4

Results

4.1 Simulation Setup

A few different test circuits are simulated to compare the state variable wave technique against that of the regular state variable based harmonic balance within fREEDA. The comparison focuses mainly on the number of iterations, execution time and memory consumption. The details of the various simulations using the wave approach include the initial proposed wave HB, wave HB with the use of capacitors ('pseudo transient'), extrapolation and gradient descent. The use of wave HB as a suitable preconditioner for more complex circuits is also examined.

The simulation setup options include the solution tolerance, tolerance stepping, iteration limits, transmission line characteristic impedance, parallel capacitor conductance value, update scaling factor, gradient descent setup and extrapolation parameters. The simulation setup options and settings are defined in Table 4.1.

The stop criteria for simulations is based on the specified solution tolerance set in the netlist file. When the error of the function defined in Section 3.2 has reach a value less than the set solution tolerance the simulation is stopped and results saved.

The wave HB approach is demonstrated with a number of circuits. The initial test circuit is a simple resistor diode used to confirm program operation. Once the simple diode circuit solved correctly other circuits tested are, full wave rectifier, charge pump, MESFET amplifier,

MMIC amplifier, soliton line and multi-soliton line. The netlists of the circuits are given in Appendix A.

	Circuit Title or Simulation Description	Section Explained in Thesis
Number Harmonics	The Number of Harmonics Considered in Calculation	Section 2.4
Solution Tolerance	The Required Solution Tolerance	Section 3.2.2
Tolerance Stepping	Use Tolerance Stepping 1=Yes 0=No	Section 3.4.4
Max Port Iterations	Maximum Number of Newton Iterations for Each NL Device	Section 3.2.2
Zref	Transmission Line	Section 3.2.1
Update Scaling Factor	The Value of Scaling Factor α	Section 3.2.2
Max NL Iterations	Maximum Iterations of Entire Circuit Loop	Section 3.2.2
Use Caps	Use Pseudo Transient 1=Yes 0=No	Section 3.4.1
Conductance Value	Conductance Value of Parallel Capacitor (G)	Section 3.4.1
Max Cap Iterations	Maximum Iterations of Pseudo Transient Loop	Section 3.4.1
Iterated Timing Analysis	Use Iterated Timing Analysis 1=Yes 0=No	Section 3.4.1
Use Extrapolation	Use Extrapolation 1=Yes 0=No	Section 3.4.2
Extrapolation Start Point	Tolerance to Start Extrapolation	Section 3.4.2
Number of Extrapolation Samples	Number of Extrapolation Samples Used in Calculation	Section 3.4.2
Use Cap Extrapolation	Use Extrapolation on Pseudo Transient 1=Yes 0=No	Section 3.4.2
Cap Extrapolation Start Point	Tolerance to Start Extrapolation on Pseudo Transient	Section 3.4.2
Number of Cap Extrapolation Samples	Number of Samples Used on Pseudo Transient Extrapolation	Section 3.4.2
Use Gradient Descent	Use Gradient Descent 1=Yes 0=no	Section 3.4.3
Gradient Descent Start	Tolerance to Start Gradient Descent Method	Section 3.4.3
Execution Time(s)	Simulation Execution time	
Number of NL Iterations	Number of Iterations of Entire Circuit Loop	
Number of Cap Iterations	Number of Pseudo Transient Iterations	
Memory Usage	Computer Memory Required to Perform Calculations	

Table 4.1 Simulation Setup Table Definitions

4.2 Simple Diode Circuit

The simple diode circuit shown in Fig. 4.1 is used as an easy proof of concept and code verification test. Being a circuit with only one nonlinear port and a well known solution

waveform it became a useful troubleshooting circuit tool.

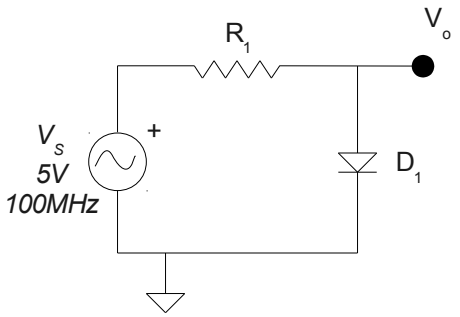

Fig. 4.1 Resistor Diode Circuit

4.2.1 Diode Simulation Setup and Results

A large number of simulations were performed on the diode circuit, the first examining the effect of varying the transmission line characteristic impedance. Two characteristic impedances are used for comparison. The various simulation setup parameters are shown in Table 4.2 along with the Newton-based state variable HB (NBSV-HB) results as a reference.

Simulation Type	Wave HB (Z_{ref}=800)	Wave HB (Z_{ref}=50)	NBSV-HB
Number Harmonics	15	15	15
Solution Tolerance	1.00E-008	1.00E-008	1.00E-008
Tolerance Stepping	0	0	NA
Max Port Iterations	100	100	NA
Zref	800	50	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1000	NA
Execution Time(s)	0.62	7.94	0.04
Number of NL Iterations	10	161	33
Number of Cap Iterations	NA	NA	NA
Memory Usage	496k	496k	NA

Table 4.2 Wave HB Simulation Setup and Results

Changing the characteristic impedance has an effect on the convergence rate of solution. The error as a function of iterations using a 50Ω and an 800Ω characteristic impedance transmission line are shown in Fig. 4.2. The number of iterations and therefore the execution

time is extended considerably with the changing of characteristic impedance from 800Ω to 50Ω.

The wave HB simulation takes 10 total iterations compared to the 33 iterations taken by the Newton-based HB simulation. The reason that the wave HB approach is still slower is because the Jacobian matrix is calculated for every iteration. The Newton-based HB only recalculates the Jacobin matrix when the iterations move far enough away from the point where the last Jacobian calculated becomes invalid. For comparison the Jacobian was only calculated twice during the Newton-based HB simulation.

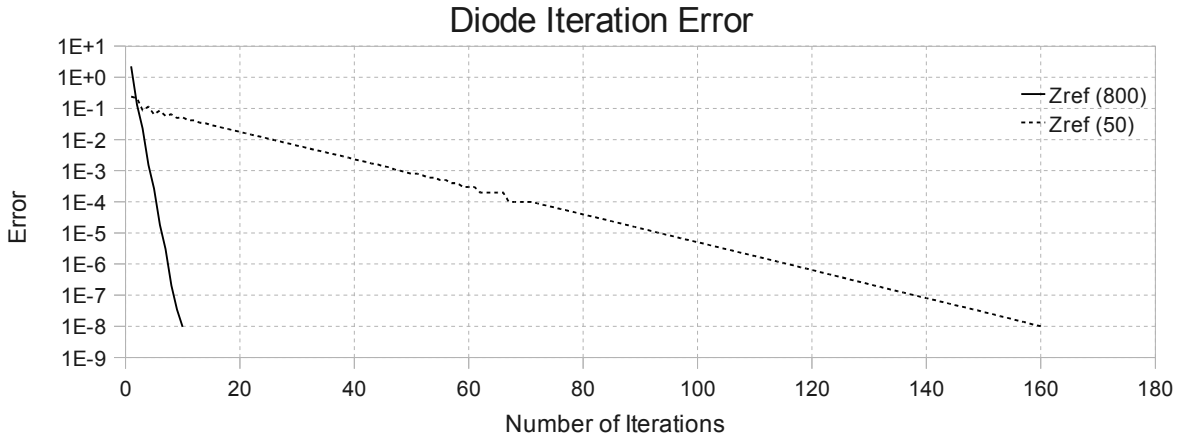

Fig. 4.2 Comparison of Different Characteristic Impedances Effect on Iterations for Resistor Diode Circuit

The use of tolerance stepping is also tested (Fig. 4.3). Tolerance stepping starts at a solution tolerance of 0.01 and steps down by a factor of 0.1 until the specified solution tolerance is achieved. Starting the simulation at a looser tolerance improves iteration speed especially at the beginning when solving each nonlinear port to a tight tolerance is an inefficiency use of processing time as the global reflection equations are still far from solution.

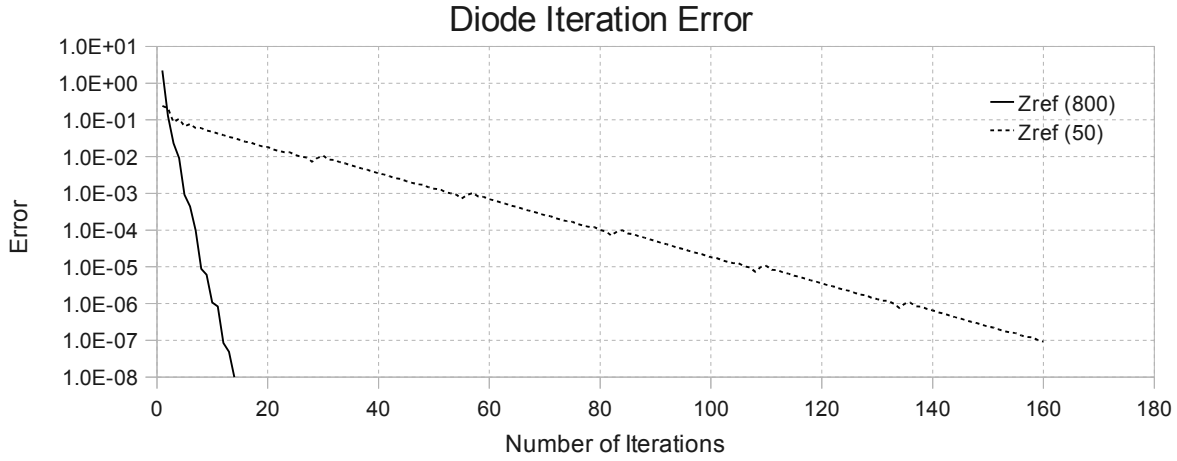

Fig. 4.3 Comparison of Different Characteristic Impedances using Tolerance Stepping

Tolerance stepping increases the total number of iterations to achieve solution. However the average time per iteration is less therefore leading to an overall processing time considerably shorter as seen in Table 4.3.

Simulation Type	Wave HB $\overline{w}$ Tol Step (Z_{ref}=800)	Wave HB $\overline{w}$ Tol Step (Z_{ref}=50)	NBSV-HB
Number Harmonics	15	15	15
Solution Tolerance	1.00E-008	1.00E-008	1.00E-008
Tolerance Stepping	1	1	NA
Max Port Iterations	100	100	NA
Zref	800	50	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1000	NA
Execution Time(s)	0.25	1.72	0.04
Number of NL Iterations	14	186	33
Number of Cap Iterations	NA	NA	NA
Memory Usage	496k	496k	NA

Table 4.3 Tolerance Stepping Setup

The resulting waveform of the voltage across the diode from each converging simulation resulted in the same solution output shown in Fig. 4.4.

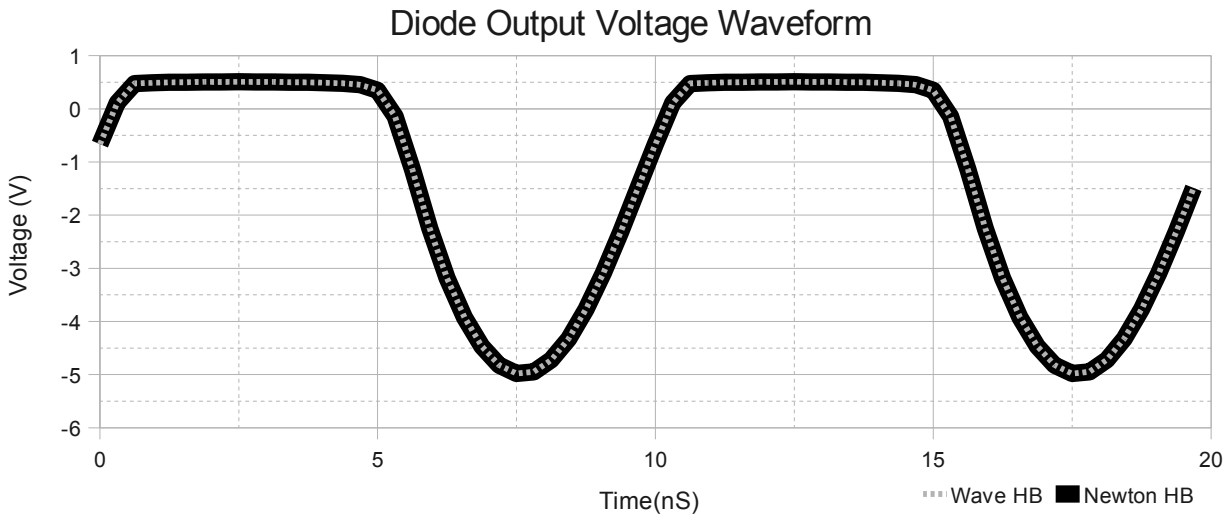

Fig. 4.4 Simulated Voltage Waveform Across Diode

As discussed in Section 3.4.1 the use of capacitors connected in parallel to the ports is attempted to improve convergence. The resulting iteration error progress of testing the 'pseudo transient' on the diode circuit is shown in Fig. 4.5. Three different simulation iteration error trends are shown in Fig. 4.5. The different simulation setup configurations and results are recorded in Table 4.4.

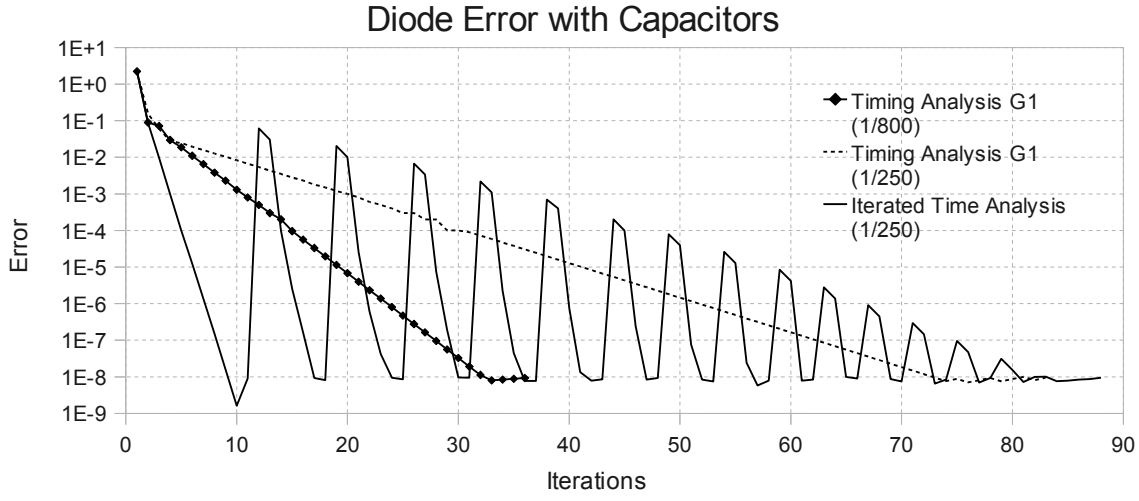

Fig. 4.5 Iteration Error Function of Resistor Diode Circuit using Capacitors

Simulation Type	Timing Analysis G_1(1/250)	Timing Analysis G_1(1/800)	Iterated Timing Analysis	NBSV-HB
Number Harmonics	15	15	15	15
Solution Tolerance	1.00E-008	1.00E-008	1.00E-008	1.00E-008
Max Port Iterations	100	100	100	NA
Zref	800	800	800	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Caps	1	1	1	NA
Conductance Value	1/250	1/800	1/250	NA
Max Cap Iterations	1000	1000	1000	NA
Iterated Timing Analysis	0	0	1	NA
Execution Time(s)	3.71	1.68	6.86	0.04
Number of NL Iterations	1	1	2.88	33
Number of Cap Iterations	82	35	81	NA
Memory Usage	496k	496k	496k	NA

Table 4.4 Simulation Setup and Results

Adjusting the value of the parallel connected conductance (G_1) is tested at $1/250\text{S}$ and $1/800\text{S}$. The use of timing analysis and iterative timing analysis as discussed in Section 3.4.1 is tested at a fixed conductance value of $1/250\text{S}$. Lowering the conductance value is equivalent to connecting a smaller value capacitor. The number of time steps required for a smaller value capacitor to charge is less when connected in parallel with the same nonlinear device therefore decreasing the number of 'pseudo transient' iterations required to reach solution.

The use of iterated timing analysis averaged 2.88 iterations of Equation (3.23) per 'pseudo transient' time step. The use of iterated timing analysis reduced the overall iterations from 82 down to 81 but with a considerable trade off of almost 3 extra sub-iterations per step. The use of iterated timing analysis with other circuits also resulted in little to no performance improvement and is therefore not shown or discussed in the remaining simulations.

Two other methods proposed to improve convergence given in Section 3.4 are extrapolation and gradient descent. The sampling for the extrapolation shown in Fig. 4.6 starts

when the residual of the error is less than 0.01. The number of samples taken before calculating the extrapolation is set to 3. Extrapolation in this simulation is performed on the reflected wave $\bar{B}$. For the gradient descent technique two thresholds of 1e-3 and 1e-5 are tested. The simulation is run normally until the error is less then the threshold and scaling of the update in accordance to the gradient begins.

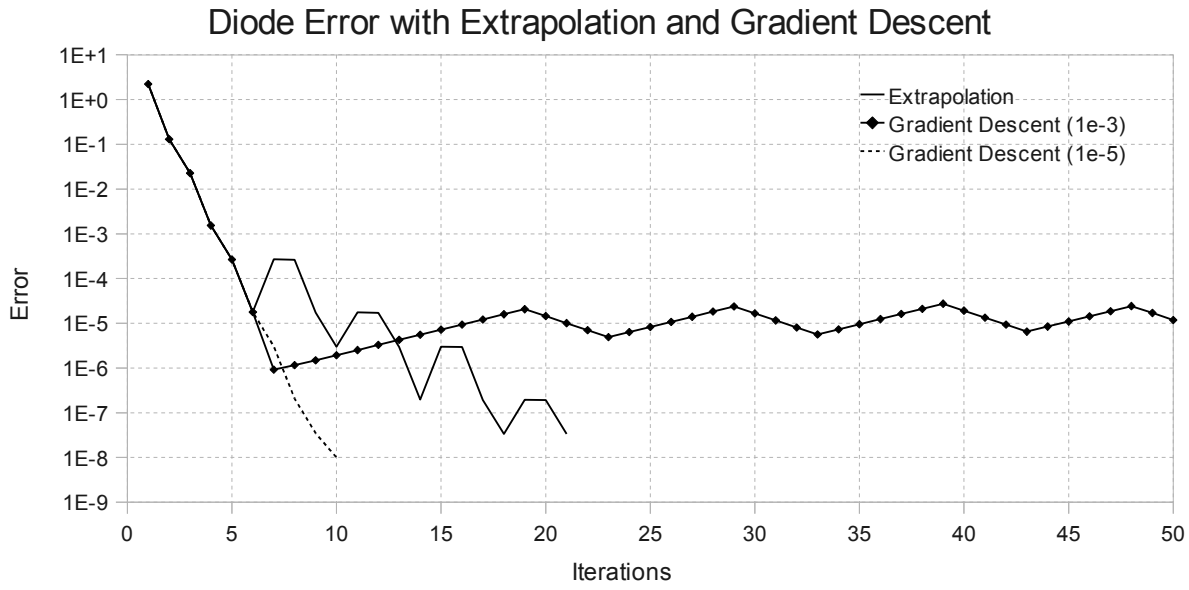

Fig. 4.6 Gradient Descent and Extrapolation Error

The extrapolation technique did not improve the convergence rate. Different numbers of samples were attempted for this circuit to analyze their effect. However the matrix template library which then calls the Lapack routine to perform the complex matrix factoring returns a failure code for samples greater than 3 for this particular circuit.

The performance of using gradient descent for this simple circuit shows similar results to the basic wave technique (Table 4.5). Setting the gradient descent activation tolerance to high results in the solution becoming trapped at an incorrect solution. Lowering the gradient descent

start tolerance prevents this from occurring and has a moderate improvement on performance near the end of the simulation.

Simulation Type	Wave HB $\bar{w}$ Extrapolation	Wave HB $\bar{w}$ Gradient Descent	Wave HB $\bar{w}$ Gradient Descent	NBSV-HB
Number Harmonics	15	10	10	15
Solution Tolerance	1.00E-008	1.00E-008	1.00E-008	1.00E-008
Tolerance Stepping	0	0	0	NA
Max Port Iterations	100	100	100	NA
Zref	800	800	800	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Extrapolation	1	NA	NA	NA
Extrapolation Start Point	1.00E-001	NA	NA	NA
Number of Extrapolation Samples	3	NA	NA	NA
Use Gradient Descent	NA	1	1	NA
Gradient Descent Start	NA	1.00E-005	1.00E-003	NA
Execution Time(s)	1.03	0.56	15.28	0.04
Number of NL Iterations	22	10	385	33
Number of Cap Iterations	NA	NA	NA	NA
Memory Usage	496k	496k	496k	NA

Table 4.5 Gradient Descent and Extrapolation Setup and Results

The final technique to be tested is the use of extrapolation on the port voltage while using 'pseudo transient'. The use of extrapolation on the port voltage has an adverse effect on the simulation. The overall effect of extrapolation on the port voltage while using capacitors severely increased the total number of iterations required and subsequently the execution time. This trend is observed with other test circuits as well. The issues that arise from the use of extrapolation appear to be related to issues in the programming code and for this reason all further extrapolation results are provided in Appendix B.

4.2.2 Diode Simulation Summary

The simulation of the simple diode circuit using wave HB was successful in the sense that

the correct solution was found. The use of tolerance stepping provides the fastest simulation which is still quite a bit slower than the Newton-based HB. The use of 'pseudo transient' also converges to the solution with a slower execution time due to the number of 'pseudo transient' time steps required to reach solution. For this simple circuit however the use the wave HB technique does not meet the performance of Newton-based HB.

4.3 Full Wave Rectifier

A full wave rectifier is commonly used to convert an alternating current containing positive and negative swings into a signal containing only positive swings. Connecting a capacitor on the output is used to filter the signal to an approximate DC signal. For the simulations the filter capacitor was removed therefore the resulting waveform is approximately a rectified sine wave. The circuit is comprised of four diodes arranged as shown in the circuit schematic Fig. 4.7 with a load resistance of R_L=10kΩ.

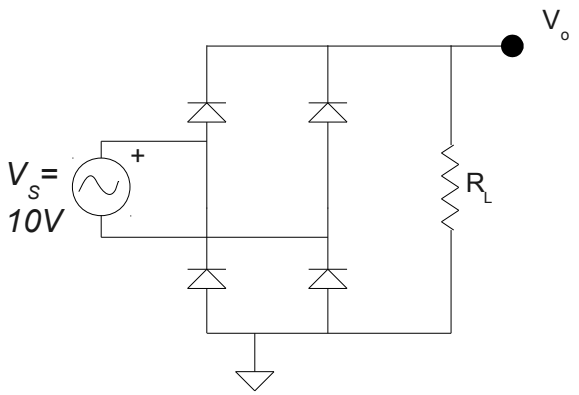

Fig. 4.7 Rectifier Schematic

4.3.1 Full Wave Rectifier Simulation Setup and Results

Similar to the diode circuit discussed previous the effect of varying the transmission line

characteristic impedance is tested. The simulation setup is shown in Table 4.6 The simulations compare the use of regular wave HB with and without the use of tolerance stepping.

Simulation Type	Wave HB (Z_{ref}=800)	Wave HB $\overline{w}$ Tol Step	NBSV-HB
Number Harmonics	25	25	15
Solution Tolerance	1.00E-007	1.00E-007	1.00E-008
Tolerance Stepping	0	1	NA
Max Port Iterations	200	200	NA
Zref	800	800	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1000	NA
Execution Time	2m29.312s	14.071s	Diverges
Number of NL Iterations	323	97	NA
Number of Cap Iterations	NA	NA	NA
Memory Usage	892k	892k	1000k

Table 4.6 Wave HB Simulation Setup

Tolerance stepping for this particular circuit improved both the convergence and execution speed. Tolerance stepping reduces the number of iterations by almost a factor of 3 and improves convergence speed by a factor of roughly 12.5.

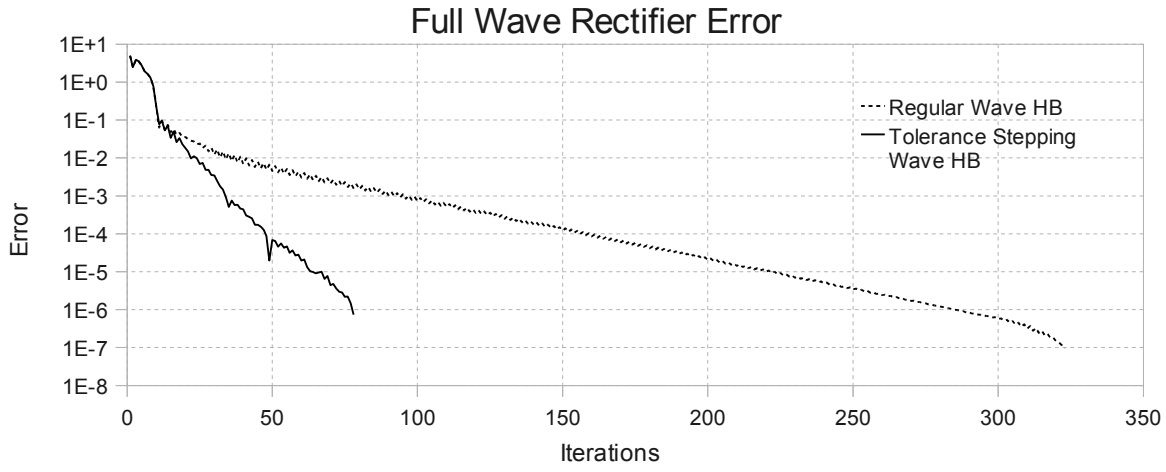

Fig. 4.8 Wave HB Error as Function of Iterations

The simulated voltage output from the rectifier circuit for a convergent solution is shown in Fig. 4.9.

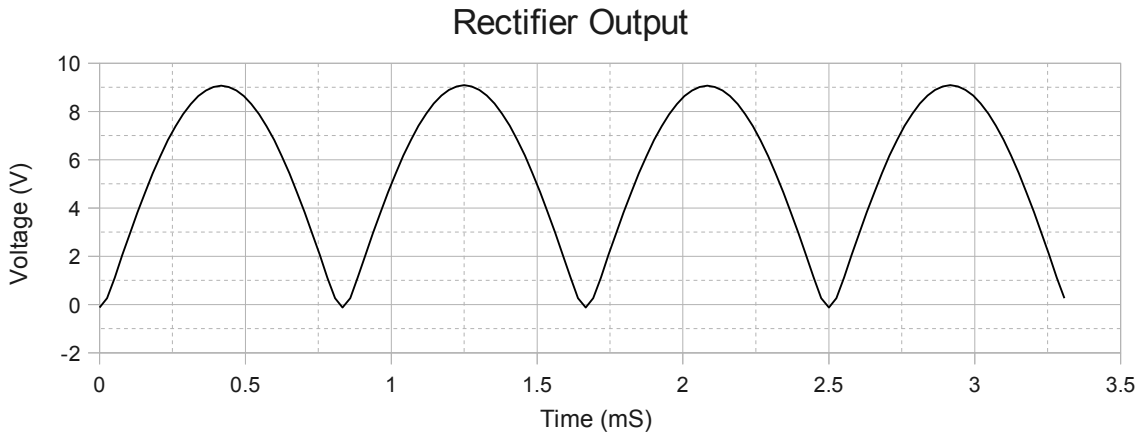

Fig. 4.9 Simulated Voltage Output Waveform (Wave HB)

The simulation setup and results from the use of 'pseudo transient' are compared to Newton-based HB and wave HB, shown in Table 4.7 and Fig. 4.10.

Simulation Type	Wave HB (Z_{ref}=800)	Wave HB $\overline{w}$ Tol Step	Wave HB $\overline{w}$ Pseudo Transient	NBSV-HB
Number Harmonics	25	25	25	25
Solution Tolerance	1.00E-007	1.00E-007	1.00E-007	1.00E-007
Tolerance Stepping	0	1	0	NA
Max Port Iterations	200	200	200	NA
Zref	800	800	800	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Caps	0	0	1	NA
Conductance Value	NA	NA	1/250	NA
Max Cap Iterations	NA	NA	1000	NA
Iterated Timing Analysis	NA	NA	0	NA
Use Extrapolation	0	0	0	NA
Execution Time	2m29.312s	14.07	1m16.68s	Diverges
Number of NL Iterations	323	97	1	NA
Number of Cap Iterations	NA	NA	230	NA
Memory Usage	892k	892k	892k	1000k

Table 4.7 Pseudo Transient Compared with Wave and Newton-based HB

Pseudo transient improved the convergence speed of wave HB but not enough to beat the tolerance stepping approach.

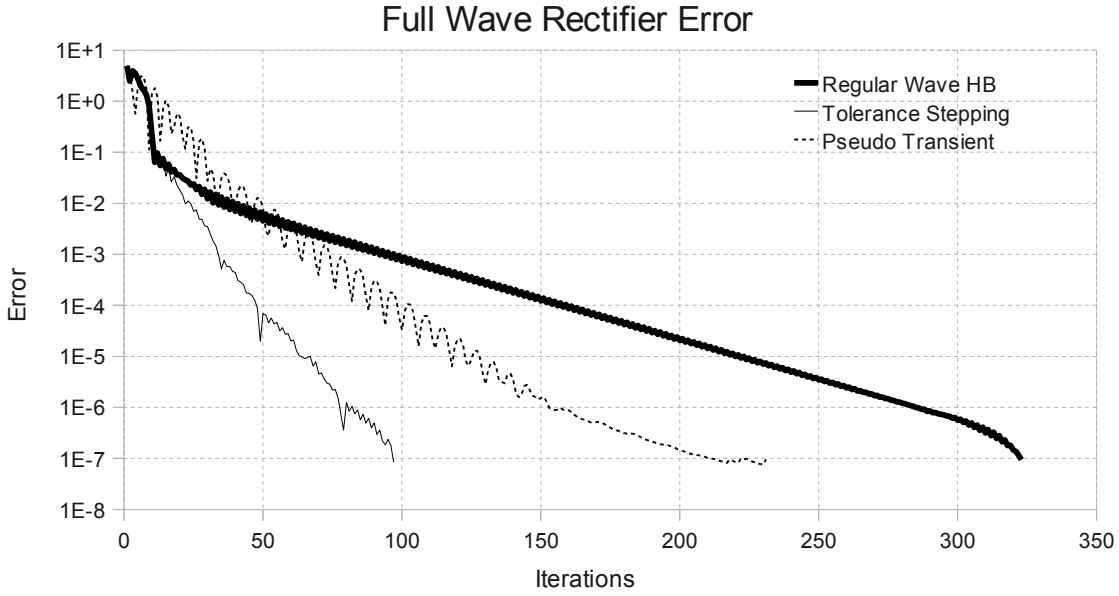

Fig. 4.10 Pseudo Transient Compared with Wave and Newton-based HB

The simulation setup to test the use of gradient descent to improve convergence is shown in Table 4.8.

Simulation Type	Wave HB $\overline{w}$ Gradient Descent	Wave HB $\overline{w}$ Gradient Descent	Wave HB $\overline{w}$ Gradient Descent	NBSV-HB
Number Harmonics	25	25	25	25
Solution Tolerance	1.00E-007	1.00E-007	1.00E-007	1.00E-008
Tolerance Stepping	0	0	1	NA
Max Port Iterations	200	200	200	NA
Zref	800	800	800	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	300	1000	1000	NA
Use Gradient Descent	1	1	1	NA
Gradient Descent Start (SF=0.1)	1.00E-005	5.00E-007	5.00E-007	NA
	No Convergence			
Execution Time	4m16.647s	2m12.414	12.808s	Diverges
Number of NL Iterations	300	288	81	NA
Number of Cap Iterations	NA	NA	NA	NA
Memory Usage	892k	892k	892k	1000k

Table 4.8 Gradient Descent Setup and Results

The use of gradient descent improved the solution time of wave HB with and without tolerance stepping. As can be seen from the gradient descent simulation starting at a tolerance of

1e-5 the gradient descent diverges from solution (Fig. 4.11). This is an example of what happens if the gradient descent is started too far away from the correct solution.

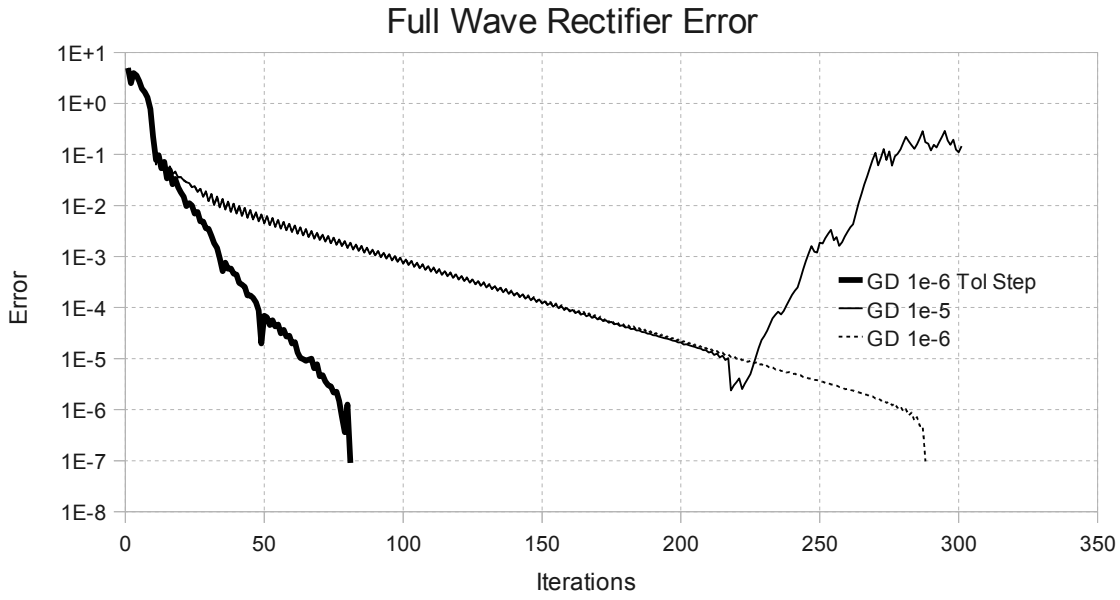

Fig. 4.11 Gradient Descent Error

Activating the gradient descent method at 1e-6 improves convergence and simulation time. The fastest simulation for solving the full wave rectifier is the combination of wave HB with tolerance stepping and gradient descent.

4.3.2 Full Wave Rectifier Simulation Summary

The full wave rectifier circuit converged to solution using wave HB with and without capacitors when the characteristic impedance was set to 800Ω. When the impedance was lowered however convergence did not occur without the use of 'psuedo transient'. Adding the use of gradient descent together with tolerance stepping proved to be the quickest simulation. For this particular circuit Newton-based HB did not converge to solution so no comparison was available as a reference.

4.4 Charge Pump

A simple charge pump circuit uses a configuration of capacitors and diodes to turn an applied sin wave input into a DC voltage output higher than the applied AC voltage. The magnitude of the output voltage is related to the number of capacitor and diode stages in the circuit. The greater the number of stages the higher the output voltage. The circuit schematic shown in Fig. 4.12 is a typical charge pump circuit.

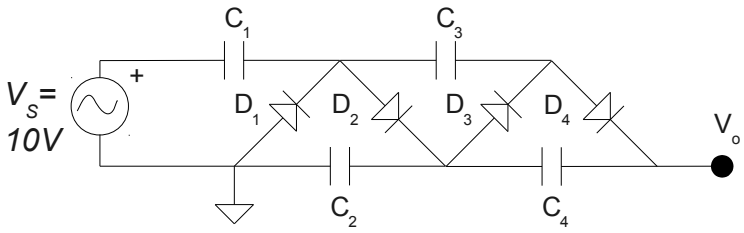

Fig. 4.12 Charge Pump Schematic

4.4.1 Charge Pump Simulation Setup and Results

The simulated voltage output waveform for a convergent simulation is shown in Fig. 4.13.

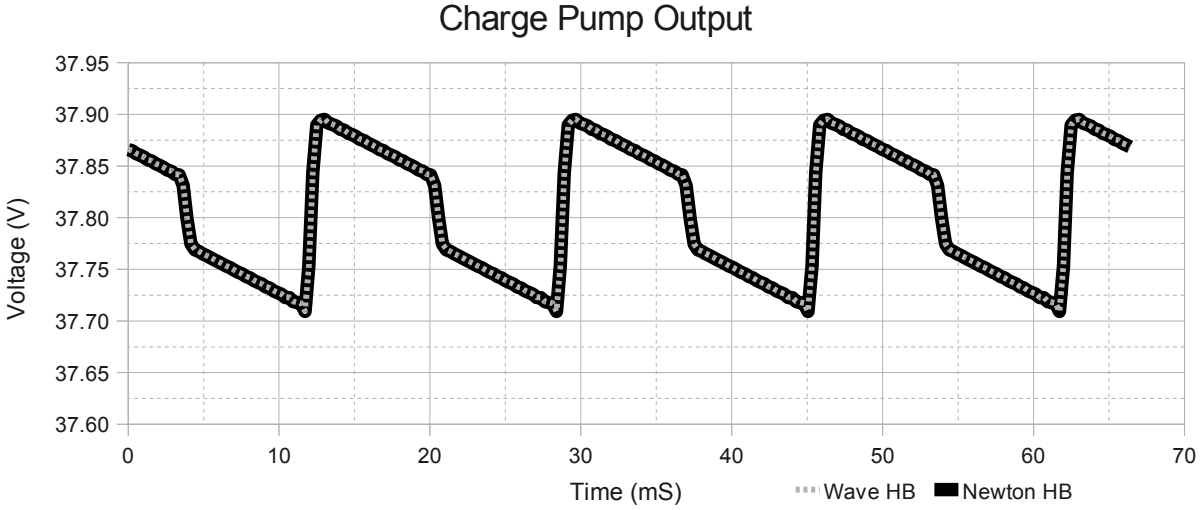

Fig. 4.13 Charge Pump Voltage Output Waveform

The applied AC voltage in all simulations of the charge pump is 10VAC and the output is loaded with a 1MΩ resistor. The simulation setup for the first comparison is the same as the full wave rectifier. The characteristic impedance is varied between 50Ω and 800Ω and the use of 'pseudo transient' with the addition of parallel capacitors is tested (Table 4.9).

Simulation Type	Wave HB (Z_{ref}=50)	Wave HB (Z_{ref}=800)	Wave HB $\overline{w}$ Tol Step	Wave HB $\overline{w}$ Pseudo Transient	NBSV-HB
Number Harmonics	25	25	25	25	15
Solution Tolerance	1.00E-005	1.00E-005	1.00E-005	1.00E-005	1.00E-005
Tolerance Stepping	0	0	1	0	NA
Max Port Iterations	200	200	200	200	NA
Zref	50	800	800	800	NA
Update Scaling Factor	0.3	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	2000	1000	NA
Use Caps	0	0	0	1	NA
Conductance Value	NA	NA	NA	1/250	NA
Max Cap Iterations	NA	NA	NA	1000	NA
Iterated Timing Analysis	NA	NA	NA	0	NA
Use Extrapolation	0	0	0	0	NA
	Iteration Limit Hit	Iteration Limit Hit			
Execution Time	8m 42s	19m12s	4m45.194s	2m25.143s	2.881s
Number of NL Iterations	1000	1000	1634	1	1000
Number of Cap Iterations				472	NA
Memory Usage	904k	904k	904k	904k	1700k

Table 4.9 Simulation Setup and Results

The charge pump circuit simulation using wave HB hit the maximum iteration limit before converging to the required solution tolerance. The error trends in Fig. 4.14 show that eventual convergence to solution is likely however the number of iterations and execution time appear to be great. The convergence improves considerably when the use of 'pseudo transient' is implemented. Unlike the previous simulations the use of the lower valued characteristic impedance of 50Ω shows better convergence characteristics.

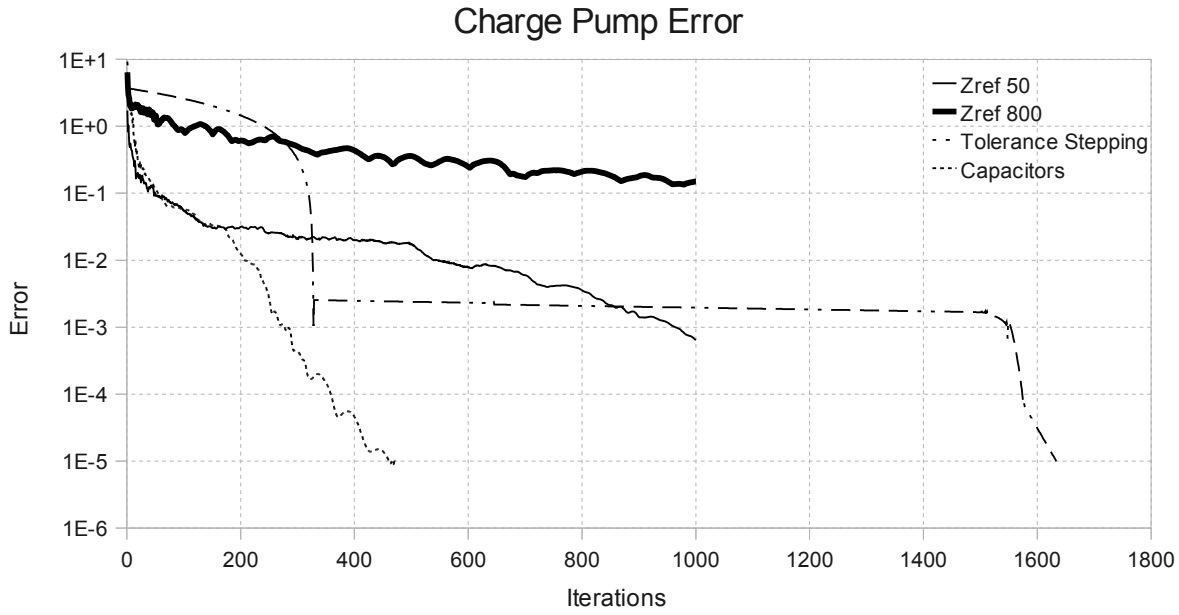

Fig. 4.14 Wave HB, Tolerance Stepping and Pseudo Transient Iteration Error Results

For this simulation 'pseudo transient' provides the best wave HB convergence results. The use of tolerance stepping also improves the convergence of regular wave HB. When compared to the reference simulation using Newton-based HB the simulation times are still much greater.

The use of extrapolation and gradient descent are attempted to improve the overall simulation further. The simulation error as a function of iterations is shown in Fig. 4.15 and is compared with that of 'pseudo transient'.

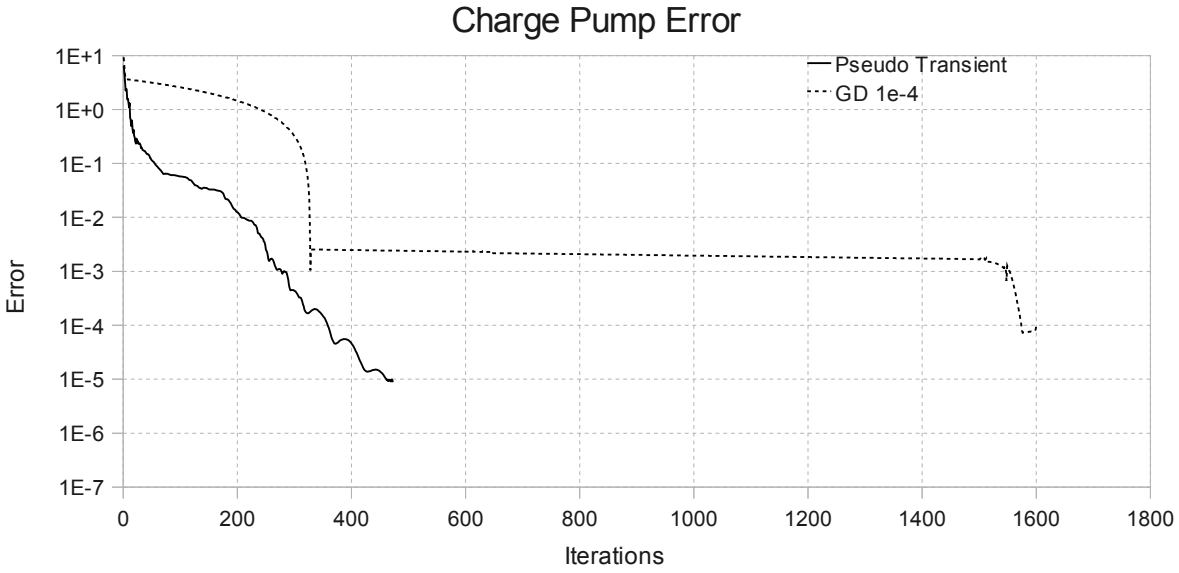

Fig. 4.15 Extrapolation and Gradient Descent Error Compared to Pseudo Transient

The simulation setup and results for comparing extrapolation and gradient descent to 'pseudo transient' are shown in Table 4.10. The use of gradient descent did not improve the performance and caused the simulation to hit the maximum iterations threshold.

Simulation Type	Wave HB $\overline{w}$ Pseudo Transient	Wave HB $\overline{w}$ Gradient Descent	NBSV-HB
Number Harmonics	25	25	15
Solution Tolerance	1.00E-005	1.00E-005	1.00E-005
Tolerance Stepping	0	1	NA
Max Port Iterations	200	200	NA
Zref	50	50	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1600	NA
Use Caps	1	NA	NA
Conductance Value	1/250	NA	NA
Max Cap Iterations	1000	NA	NA
Iterated Timing Analysis	0	NA	NA
Use Gradient Descent	NA	1	NA
Gradient Descent Start	NA	1.00E-004	NA
Execution Time	2m25.143s	4m51.04s	2.881s
Number of NL Iterations	1	1600	1000
Number of Cap Iterations	472	NA	NA
Memory Usage	904k	904k	1700k

Table 4.10 Extrapolation and Gradient Descent Compared to 'Pseudo Transient'

4.4.2 Charge Pump Simulation Summary

The results from the charge pump circuit are poor when compared to the reference simulation using the Newton-based HB approach. The use of 'pseudo transient' showed the best performance out of the wave HB techniques. Tolerance stepping also produced an execution time similar to 'pseudo transient' but required many iterations to achieve convergence.

4.5 MESFET Amplifier

So far the circuits tested with the wave HB approach have consisted of diodes and passive components. The MESFET amplifier circuit is the first circuit using an active element with more than one port per device. The schematic for the MESFET amplifier is shown in Fig. 4.16.

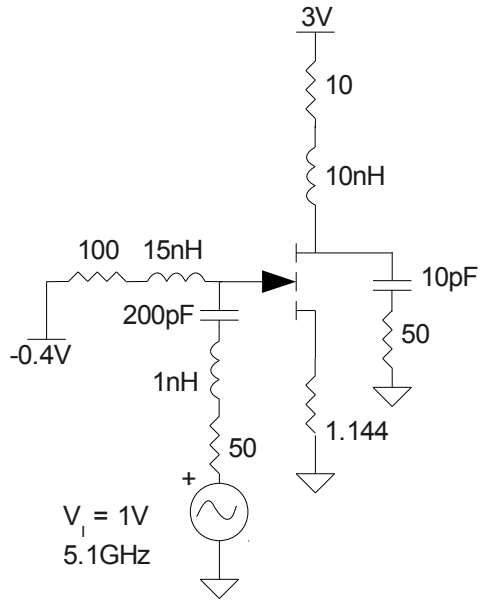

Fig. 4.16 MESFET Amplifier Schematic

The simulated solution waveform for the MESFET amplifier output is shown in Fig. 4.17.

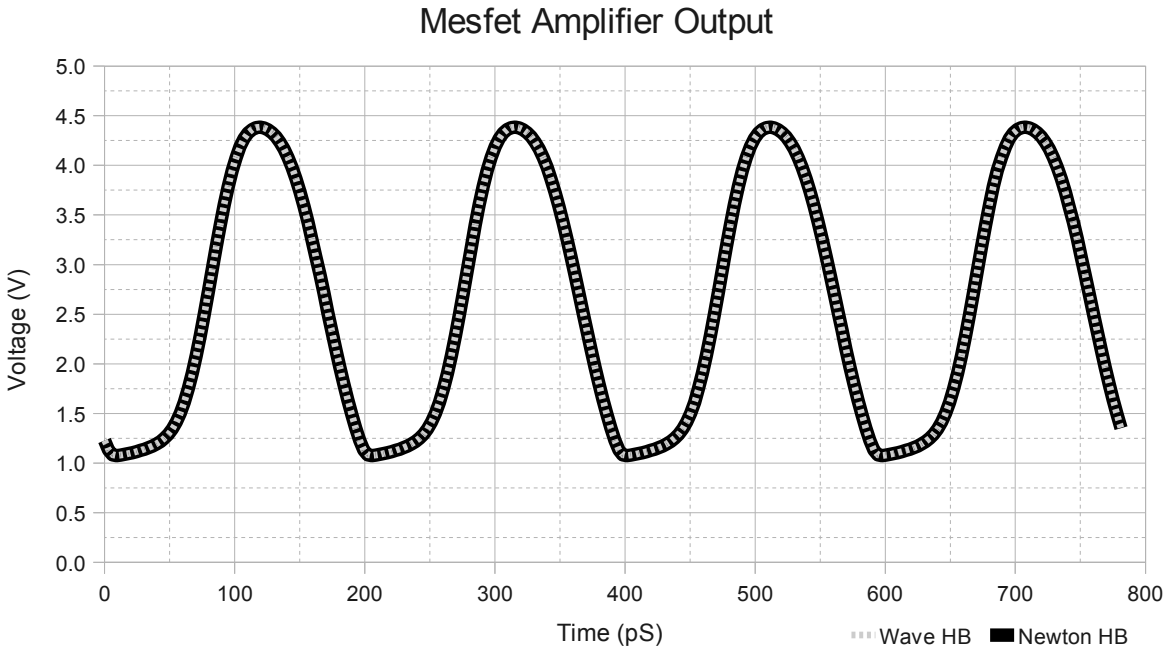

Fig. 4.17 Simulated Solution Waveform

The first simulation setup is shown in Table 4.11 using the same variation as the previous circuits tested examining the use of different characteristic impedances at 50Ω and 800Ω.

Simulation Type	Wave HB (Z_{ref}=50)	Wave HB (Z_{ref}=800)	NBSV-HB
Number Harmonics	20	25	25
Solution Tolerance	1.00E-006	1.00E-006	1.00E-006
Tolerance Stepping	0	0	NA
Max Port Iterations	200	200	NA
Zref	50	800	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1000	NA
Execution Time(s)	5.637s	44.816s	.076s
Number of NL Iterations	26	96	NA
Number of Cap Iterations	NA	NA	NA
Memory Usage	1000k	1000k	1020k

Table 4.11 MESFET Amplifier Wave HB Simulation Setup and Results

Similar to the simulations before changing the value of the characteristic impedance has an effect on the convergence rate and therefore the number of iterations. For the MESFET however the use of the lower 50Ω impedance instead of 800Ω displayed better convergence

characteristics (Fig. 4.18).

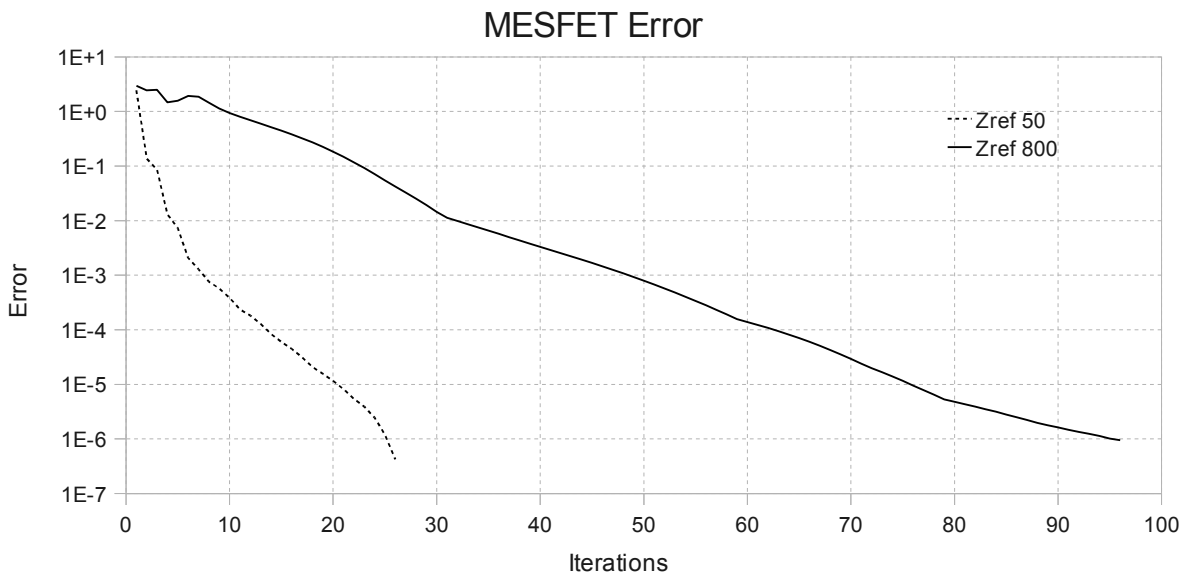

Fig. 4.18 Characteristic Impedance Effect on Convergence

The use of tolerance stepping and 'pseudo transient' techniques is tested next. The tolerance stepping approach is run twice using a different number of maximum iterations for the nonlinear ports. The first simulation use a maximum of 200 iterations to solve for each reflected power wave and the other simulation uses 1. The simulation setup and results are shown in Table 4.12.

Simulation Type	Wave HB $\overline{w}$ Tol Step '200' Port Iteratios	Wave HB $\overline{w}$ Tol Step '1' Port Iterations	Wave HB $\overline{w}$ Pseudo Transient	NBSV-HB
Number Harmonics	20	25	25	25
Solution Tolerance	1.00E-006	1.00E-006	1.00E-006	1.00E-006
Tolerance Stepping	1	1	0	NA
Max Port Iterations	200	1	200	NA
Zref	50	50	50	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Caps	0	0	1	NA
Conductance Value	NA	NA	1/250	NA
Max Cap Iterations	NA	NA	1000	NA
Iterated Timing Analysis	NA	NA	0	NA
Execution Time(s)	6.461s	1.057s	6.249s	.076s
Number of NL Iterations	20	107	1	NA
Number of Cap Iterations	NA	NA	21	NA
Memory Usage	1000k	1000k	1000k	1020k

Table 4.12 MESFET Amplifier Tolerance Stepping Simulation Setup and Results

The use of 'pseudo transient' and tolerance stepping using a maximum of 200 iteration per nonlinear port produced very similar simulation results. Taking just over 6 seconds to converge to solution in 20 iterations. Tolerance stepping with only one iteration per nonlinear device required over 100 iterations to converge to solution however the execution time was considerably lower at just over 1 second (Fig. 4.19).

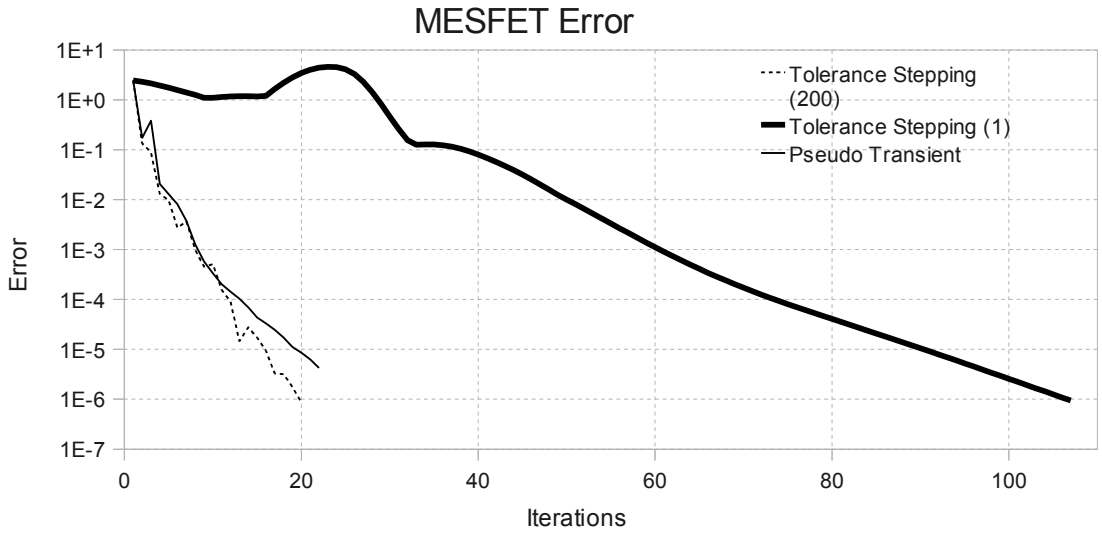

Fig. 4.19 Simulation Iteration Error

4.5.2 MESFET Amplifier Simulation Summary

The MESFET amplifier differs from the other circuits tested because it contains an active device which contains two ports per device. The use of tolerance stepping and only one iteration per port produces the quickest solution time. The use of one port iteration per nonlinear device saves time because only one small matrix decomposition is required per global nonlinear iteration. For this simulation the effects are greater still because a device with two ports doubles the matrix and vector sizes within the nonlinear port routine. This accounts for the substantial time savings caused by reducing the maximum number of port iterations from 200 down to 1. The other simulation techniques preformed comparable to the other test circuits, including extrapolation causing an adverse effect on the solution convergence. The use of gradient descent was not preformed due to the implementation being coded to support devices with only one state.

4.6 Soliton Line

So far the circuits that have been tested consist of very few nonlinear elements and do not require a large number of harmonics to accurately describe the solution waveform. The soliton line however is a highly nonlinear network that consists of many nonlinear devices and requires a large number of harmonics to accurately represent the solution. The schematic of the soliton line is shown in Fig. 4.20. It consists of a high frequency source driving 47 diodes connected to each other by transmission lines and terminated with a 50Ω load [28,29].

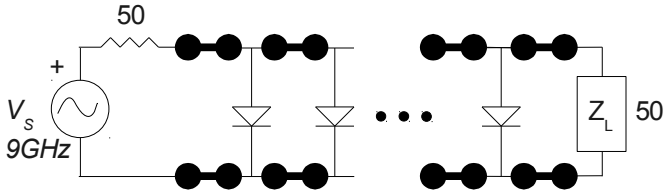

Fig. 4.20 Soliton Line Schematic

The output voltage waveform at the end of the soliton line is shown in Fig. 4.21. It can be seen that the waveform exhibits strong nonlinearities. The size of the system to be solved for this

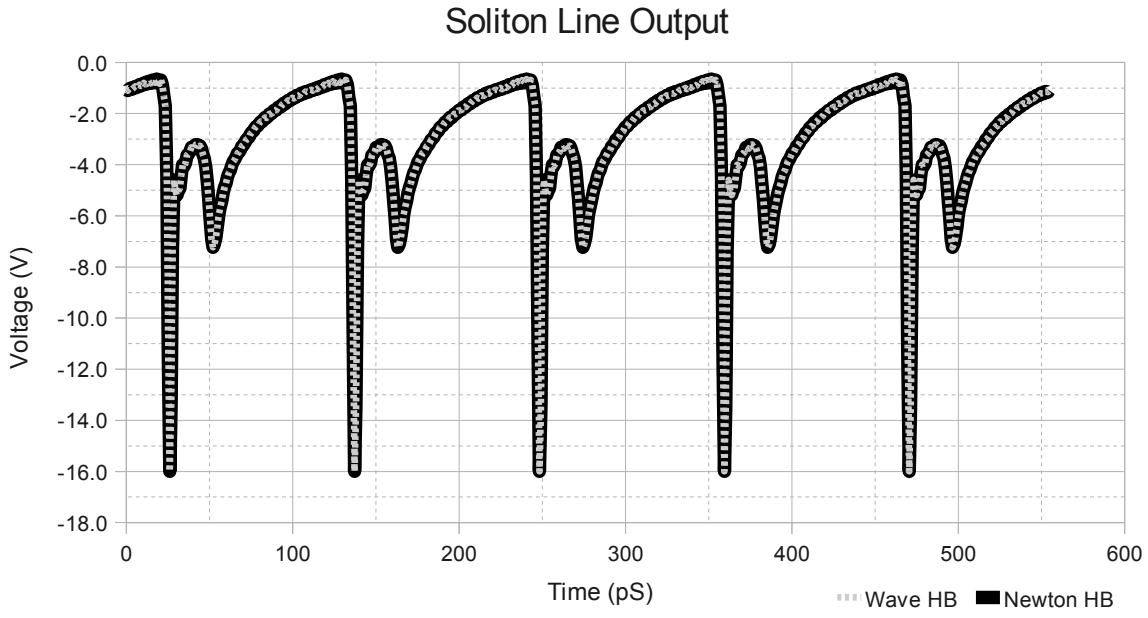

Fig. 4.21 Soliton Line Voltage Output Waveform

circuit using Newton-based HB consist of a square matrix, having a dimension of 3807 x 3807.

4.6.1 Soliton Line Simulation Setup and Results

The soliton line is first solved to a loose tolerance of 1e-2 to compare the performance between the various different methods covered so far. The reason for solving to a loose tolerance is due to the fact that simulation of the soliton line takes a considerable amount of time to solve. The first simulation is used to compare the effect of different characteristic impedances. The use of 50Ω and 800Ω is compared in Table 4.13.

Simulation Type	Wave HB (Z_{ref}=50)	Wave HB (Z_{ref}=800)
Number Harmonics	40	40
Solution Tolerance	1.00E-002	1.00E-002
Max Port Iterations	100	100
Zref	50	800
Update Scaling Factor	0.3	0.3
Max NL Iterations	100	1000
		Iteration Limi Hit
Execution Time	41m2.025s	82m4.09s
Number of NL Iterations	75	101
Number of Cap Iterations	NA	NA
Memory Usage	13.2M	13.2M

Table 4.13 Characteristic Impedance Comparison

The use of a 50Ω characteristic impedance shows better solution performance and is therefore used for all remaining simulations. The error as a function of iterations using the two different impedances is shown in Fig. 4.22.

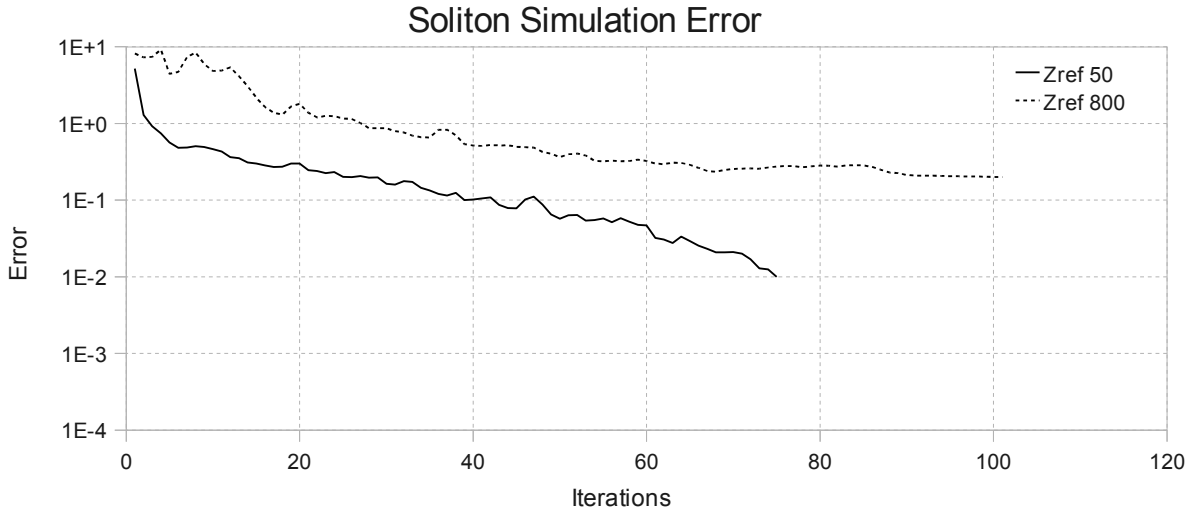

Fig. 4.22 Characteristic Impedances Effect on Convergence

The use of 'pseudo transient' is tested at different tolerances. As the tolerance is increased the number of iterations and the amount of time required to reach solution increases. Three different simulations are run with various tolerances as shown in Table 4.14. The Newton-based HB approach is also included for comparison.

Simulation Type	Wave HB $\overline{w}$ Pseudo Transient	Wave HB $\overline{w}$ Pseudo Transient	Wave HB $\overline{w}$ Pseudo Transient	NBSV-HB
Number Harmonics	40	40	40	40
Solution Tolerance	1.00E-002	1.00E-003	1.00E-006	1.00E-005
Max Port Iterations	100	100	100	NA
Zref	50	50	50	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Caps	1	1	1	NA
Conductance Value	1/250	1/250	1/250	NA
Max Cap Iterations	1000	1000	1000	NA
Iterated Timing Analysis	0	0	0	NA
				Source Step
Execution Time	28m7.471s	74m4.636s	301m15s	57m51.676s
Number of NL Iterations	1	1	1	354
Number of Cap Iterations	75	114	248	NA
Memory Usage	13.2M	13.2M	13.2M	340.1M

Table 4.14 Different Tolerance Comparisons

A couple of points of interest from the results in Table 4.14. The total amount of memory required to perform the solution calculation is substantially lower in the wave HB approach, approximately 26 times less then Newton-based HB. The Newton-based HB approach also uses a source-stepping technique. The source stepping technique is the same as solving the circuit at a fraction of the input voltage and considering less harmonics. At the next step the voltage and number of harmonics is increased until the voltage is at the specified amount and all harmonics are considered. Without source stepping the Newton-based HB fails to converge. Another interesting observations is that Newton-based HB takes a considerable amount of time to reach a solution which is relatively close to correct. Once a solution is relatively close Newton-based HB takes very few iterations and therefore a short amount of time to solve to a tight tolerance. On the other hand wave HB reaches a solution which is close in a relatively short amount of time but takes much longer to reach a tight tolerance.

The use of tolerance stepping and using a different limit on the number of nonlinear port iterations is shown in Fig. 4.23. Increasing the limit on the Newton iterations for each device port typically extends the overall solution time. This occurs because more time is spent calculating each device ports reflections. The use of more Newton iterations for each device improves the rate and likeliness of global convergence of the system. However increasing the number past a certain point has little effect, as the Newton iterations solve to solution before the limit is hit. The major advantage of lowering the iteration limit is reducing the time spent on device ports which have slowly converging Newton iterations. This can improve the overall solution time with the trade offs being a reduction in the rate and likeliness of global convergence.

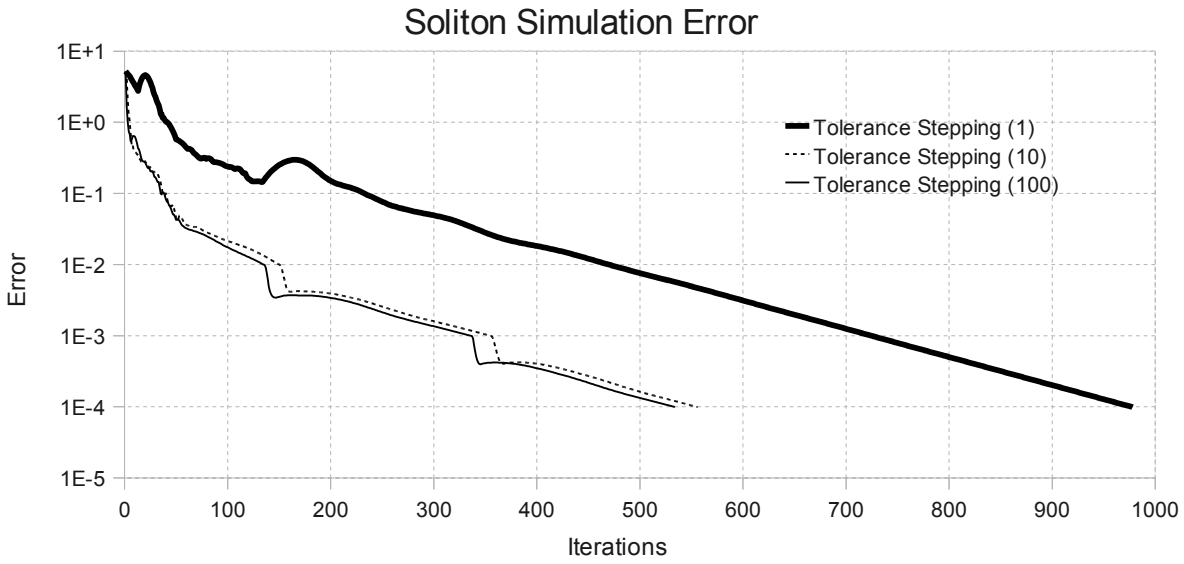

Fig. 4.23 Comparison of Tolerance Stepping and Max Port Iteration Setting

The use of tolerance stepping improves the simulation convergence time by decreasing the average time spent on each iteration at the cost of requiring more iterations. The same holds true when the limit of iterations per local device port reflection calculation is lowered. From Table 4.15 using tolerance stepping combined with only one iteration of the reflection calculation for each port provides the quickest solution time.

Simulation Type	Wave HB $\overline{w}$ Tol Step '1' Port Iteratios	Wave HB $\overline{w}$ Tol Step '10' Port Iteratios	Wave HB $\overline{w}$ Tol Step '100' Port Iteratios	NBSV-HB
Number Harmonics	40	40	40	40
Solution Tolerance	1.00E-004	1.00E-004	1.00E-004	1.00E-006
Tolerance Stepping	1	1	1	NA
Max Port Iterations	1	10	100	NA
Zref	50	50	50	NA
Update Scaling Factor	0.3	0.3	0.3	NA
Max NL Iterations	1000	1000	1000	NA
Use Caps	0	0	0	NA
				Source Step
Execution Time	41m7.773s	45m 50.66s	103m5.865s	57m51.676s
Number of NL Iterations	978	556	534	354
Number of Cap Iterations	NA	NA	NA	NA
Memory Usage	13.2M	13.2M	13.2M	340.1M

Table 4.15 Tolerance Stepping Simulation Setup and Results

In Table 4.16 a comparison is made between 'pseudo transient', tolerance stepping and Newton-based HB solved to a tight solution tolerance of 1e-6. The tolerance stepping solution simulation time is somewhat comparable to that of Newton-based HB but still comes up short.

Simulation Type	Wave HB $\overline{w}$ Pseudo Transient	Wave HB $\overline{w}$ Tol Step	NBSV-HB
Number Harmonics	40	40	40
Solution Tolerance	1.00E-006	1.00E-006	1.00E-006
Tolerance Stepping	0	1	NA
Max Port Iterations	100	10	NA
Zref	50	50	NA
Update Scaling Factor	0.3	0.3	NA
Max NL Iterations	1000	1500	NA
Use Caps	1	0	NA
Conductance Value	1/250	NA	NA
Max Cap Iterations	1000	NA	NA
Iterated Timing Analysis	0	NA	NA
			Source Step
Execution Time	301m15s	68m28s	57m51.676s
Number of NL Iterations	1	1000	354
Number of Cap Iterations	248	NA	NA
Memory Usage	13.2M	13.2M	340.1M

Table 4.16 Comparison of Wave HB and Newton-based HB

4.6.2 Soliton Line Simulation Summary

The soliton line is a relatively nonlinear circuit which presents some difficulty when solving with a harmonic balance approach. The use of wave HB combined with tolerance stepping shows performance which is similar to that of Newton-based HB but does not improve the overall time required to solve the simulation. Wave HB approach is a form of relaxation technique with a low convergence rate. This differs from the path taken by Newton-based HB which takes a considerable amount of time to get close to solution, but once there converges rapidly. The possibility of mixing the two together is present in the Section 4.7. Another advantage of the wave approach is the substantial reduction in memory usage. This becomes an even bigger savings as the size of the circuit in question grows.

4.7 Wave HB as a Preconditioner

As was mentioned in Section 4.6 the wave HB approach relaxes towards the solution. This results in a considerable simulation time for complex systems. The use of Newton-based HB has much better convergence performance once the solution is relatively close. The cost of getting relatively close to solution using Newton-based HB can be high due to the computational cost of decomposing the Jacobian matrix. For smaller circuits this task is quite quick but as the circuit and number of harmonics considered grows execution time increases dramatically. This leads to the idea of using wave HB as a preconditioner for Newton-based HB when complex circuits are simulated. In this section combining the relaxation approach of wave HB with the fast converge rate of Newton-based HB when near solution is examined. The soliton line in Section 4.6 is tested along with a circuit containing two soliton lines.

4.7.1 Soliton Line with Wave HB as a Preconditioner

For testing the wave HB as a preconditioner a simulation tolerance of 1e-2 is set and the solution is solved. The resulting solution waveform is shown in Fig. 4.24. Determining a stopping criteria for the change from one method to the other was determined using a trial and error approach. For this particular example the tolerance of 1e-2 showed the best compromise between time and accuracy.

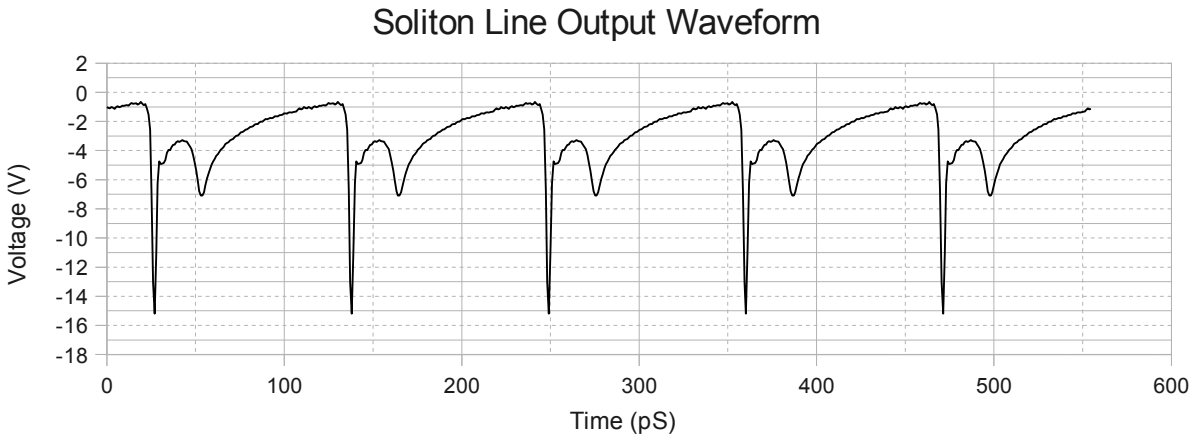

Fig. 4.24 Soliton Output Waveform 1e-2 Solution Tolerance

It can be seen from Fig. 4.24 that the simulated solution although only solved to 1e-2 is still resemblant of the correct solution waveform shown in Fig. 4.21

The resulting solution is then feed into the start of Newton-based HB as the initial guess and solved without source stepping to a tolerance of 1e-6. The results of the simulation are shown in Table 4.17.

Simulation Type	Wave HB $\overline{w}$ Tol Step	Wave HB (Z_{ref}=50)	Wave HB $\overline{w}$ Pseudo Transient	Combination of Wave HB and NBSV-HB	NBSV-HB
Number Harmonics	40	40	40	40	40
Solution Tolerance	1.00E-002	1.00E-002	1.00E-002	1.00E-006	1.00E-006
Tolerance Stepping	0	0	0	NA	NA
Max Port Iterations	1	20	1	NA	NA
Zref	50	50	50	NA	NA
Update Scaling Factor	0.3	0.3	0.3	NA	NA
Max NL Iterations	1000	100	1000	NA	NA
Use Caps	0	0	1	NA	NA
Conductance Value	NA	NA	1/250	NA	NA
Max Cap Iterations	NA	NA	1000	NA	NA
Iterated Timing Analysis	NA	NA	0	NA	NA
Preconditioner Time	NA	NA	NA	18m35s	NA
Regular HB Time	NA	NA	NA	11m46.44s	Source Step
Execution Time	18m35s	41m2.025s	28m7.471s	30m21.44s	57m51.676s
Number of NL Iterations	470	75	1	35	354
Number of Cap Iterations	NA	NA	75	NA	NA
Memory Usage	13.2M	13.2M	13.2M	340.1M	340.1M

Table 4.17 Soliton Line Wave HB as a Preconditioner Results

The use of wave HB as a preconditioner achieves a solution speed that is faster than Newton-based HB on its own. The use of one iteration per port and a solution tolerance of 1e-2 is used for the preconditioner. The result is then used as the initial guess for Newton-based HB yielding a net time of just over half an hour almost twice as fast as the reference simulation using Newton-based HB.

4.7.2 Multiple Soliton Line with Wave HB as a Preconditioner

The advantages of using the wave HB as a preconditioner should become more evident as the circuit size grows and the decomposition of the Jacobian matrix becomes more expensive. To verify this, two soliton lines are connected as shown in Fig. 4.25. The sources are set to have a 180 degree phase shift from one another during simulation.

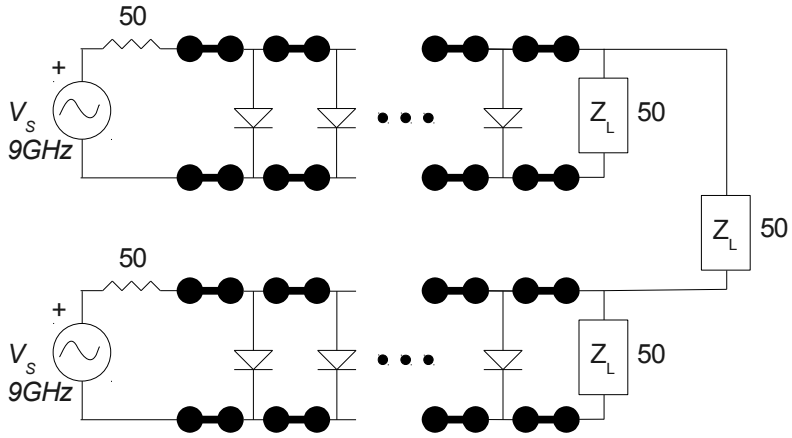

Fig. 4.25 Multiple Soliton Line Schematic

The circuit is solved using wave HB to a tolerance of 1e-2 and then used as the initial guess for Newton-based HB. Results of the preconditioner testing are shown in Table 4.18.

Simulation Type	Wave HB (Z_{ref}=50)	Combination of Wave HB and NBSV-HB	NBSV-HB
Number Harmonics	40	40	40
Solution Tolerance	1.00E-002	1.00E-006	1.00E-006
Tolerance Stepping	0	NA	NA
Max Port Iterations	1	NA	NA
Zref	50	NA	NA
Update Scaling Factor	0.3	NA	NA
Max NL Iterations	1000	NA	NA
Use Caps	1	NA	NA
Conductance Value	1/250	NA	NA
Max Cap Iterations	1000	NA	NA
Iterative time	0	NA	NA
Preconditioner Time	NA	85m38.145s	NA
Regular HB Time	NA	47m1.255s	NA
Execution Time	85m38.145s	132m39.4s	421m9.649s
Number of NL Iterations	1	15	390
Number of Cap Iterations	116	NA	NA
Memory Usage	64M	1300M	1300M

Table 4.18 Multiple Soliton Line Wave HB as a Preconditioner Results

The use of wave HB as a precondition for the multiple soliton line improves the overall solution time considerably. The total simulation time is almost 5 hours less with the use of wave HB and Newton-based HB together when compared with just Newton-based HB. Another observation is the reduction in memory required to perform the Newton-based HB calculation when compared to that of wave HB. This memory savings is however lost once the Newton-based HB is used to converge to the final solution.

4.7.3 Wave HB as a Preconditioner Summary

Wave HB as a preconditioner used as a preconditioner for Newton- harmonic balance results in shorter simulation times, at least for some circuits. The simulation time savings become more important as the circuit size grows. Wave HB uses substantial less memory when solving the system which can become an issue for the Newton-based HB technique used as a reference

throughout this work. However this benefit is lost when wave HB is used as a preconditioner.

4.8 Performance Comparisons

A variety of circuits have been tested in the previous sections. Some observations can be drawn from these simulations. Wave HB being a form of relaxation technique results in a solution process which exhibits this behavior. Relaxing to the solution can be beneficial for convergence but it also suffers from much slower solution times. Solving the reflections for each port individually is shown to conserve the amount of memory required, especially for larger and more complex circuits. The use of 'pseudo transient' showed some improvements and the ability to moderately control convergence.

If the conductance value used in 'pseudo transient' is set to zero then the simulation becomes the normal wave HB approach. Using capacitors connected to the nonlinear device ports allows the Newton iterations for each port to converge faster. There is a trade off between using 'pseudo transient' to improve convergence speed of the individual ports and the overall time that is required for the 'pseudo transient' to reach solution. In some instants this trade off results in an improved overall solution time, for others it does not.

The most promising of the techniques used are those of 'pseudo transient' and tolerance stepping. For relatively large nonlinear circuit 'pseudo transient' can be used to help convergence. For circuits of moderate size tolerance stepping seems to offer the quickest solution time. The attempts at extrapolation did not produce positive results. The problem with extrapolation appears to be in the MTL when solving the Moore Penrose pseudo inverse. Upon solving the LU decomposition Moore Penrose pseudo inverse the Lapack library returns various different

problem codes dependent on the number of extrapolation samples selected. Gradient descent did have a mild improvement however its use was only beneficial close to the end of the iteration process so its overall benefit is low.

The final use of wave HB tests its ability of being a type of preconditioner. The results of a loose tolerance solution are used as an initial guess for a conventional harmonic balance approach. The combination of the two methods has an overall improvement in the solution time of the large circuits tested. The trade off being a reduction in the size of system that can be solved due to the memory-intensive Newton-based HB method. However the harmonic balance technique provided in fREEDA does not use the Krylov-subspace method and results would be different otherwise.

Chapter 5

Conclusion

5.1 Conclusion

A rigorous extension of the multiple port reflection technique [3] for HB formulated using state variables for nonlinear circuit simulations is presented for the first time. The simulation technique does not require the use of a preconditioner or a specific initial guess. During the solution process only a small matrix decomposition is required for each port iteration. This has a considerable impact on reducing the amount of memory required during computation when compared to Newton-based HB techniques. To improve convergence the use of power waves in conjunction with parallel connected capacitors, minimum polynomial extrapolation and gradient descent methods has also been examined for the first time.

The use of power waves alone worked well for simple circuits however issues arose when attempting to solve relatively complex circuits. The addition of parallel capacitors connected to the nonlinear device ports to create a pseudo transient approach shows convergence improvements, typically at the cost of increased solution execution time. A tolerance stepping method was implemented which had a positive improvement on the time required to reach the solution.

The use of power waves in combination with minimum polynomial extrapolation showed no improvement over the regular wave HB technique. However implementation issues are suspected in the calculation of the extrapolation process resulting in unfavorable outcomes. The

problem with extrapolation is suspected to be related to the MTL and the LAPACK LU decomposition calculations. The values returned from the matrix decomposition indicate that there is a problem factoring the matrix. Further investigation is required of this calculation before extrapolation can be considered not to help with the solution execution time. The use of transient analysis simulation using power wave relaxation techniques based on state variables shows promising results when performing extrapolation [27]. It is believed that the use of extrapolation combined with the wave HB approach could improve the overall solution time and produce performance results better than the Newton-based HB approach used for comparison in this work.

A gradient descent technique was also tested with the use of power waves. The method showed minor improvement in the convergence rate once relatively close to solution, however causes problems if activated early.

The use of wave HB as a type of preconditioner for Newton-based HB shows promise when simulating relatively large nonlinear circuits. Overall the wave HB approach shows promise in some areas and introduces problems in others. There is still a large amount of investigation required to make this approach a viable alternative to the existing harmonic balance techniques. The code written to test this approach still suffers from inefficiencies both in its structure and the selected support libraries. The MTL presented many issues when dealing with complex number operations.

5.2 Future Research

There are many areas that still require investigating. For example a simulation using

'pseudo transient' has three separate iterations loops occurring. Those being at the individual nonlinear port using Newton's method, iterations of waves between the linear and nonlinear ports (3.23) and the pseudo transient time steps. The maximum number of nonlinear port iterations and the value of the parallel conductance are adjustable. Finding a correlation between the circuit being simulated and the most efficient combination could benefit the solution process.

For the Newton iterations the scaling factor is fixed in all presented simulations, changing this number has an effect on convergence and speed. Finding an optimal setting for this value for each simulation could lead to a more general performance improvement. The effect of the selected transmission line characteristic impedance also effects solution convergence and it varies circuit to circuit. Finding a relationship between the circuit and this impedance could also improve the solution convergence.

The simulations presented are mostly made up of diode circuits with the exception of the MESFET amplifier. The use of fREEDA helped with the implementation however there is not a great selection of device models within the included library. This limited the number of test circuits available. The development of new device models would lead to a greater variety of test circuits therefore could further test the worthiness of this approach. The complexities attributed to supporting circuits containing nonlinear devices with varying number of ports makes the programing quite difficult due to the various sizes and ordering of matrices and vectors. A problem is suspected with the programming for circuits containing more than one device with two or more ports. Due to time constraints this issue was not located and corrected before the preparation of this paper and further debugging is required to locate this problem.

Performance gains could also come from implementing more efficient coding and

replacing the matrix template library with one that is better tested and supports more complex number operations. This may also help with the suspected problem when using extrapolation. The wave algorithm is also setup in a way that it could possibly benefit from parallel programing, making use of modern multi-core and cluster computers.

The final area of interest is the use of the wave technique as a possible preconditioner for other harmonic balance techniques which require an initial guess close to solution before starting the simulation to achieve convergence.

Appendices

Appendix A

Appendix A.1 Simple Diode Circuit Netlist

```
*** Wave HB Simple Diode Circuit ***

*Simulation options
**********************************************************
.options freq=1e8

*Simulation Type and setup parameters
**********************************************************
.wavehb n_freqs=15  fundamental=freq tol=1e-8 usecaps=0 n_iter=100  zref=800

*Circuit netlist
**********************************************************
vsource:v1 1 0 f=freq vac=5 phase=-90

resistor:r1 1 2 r=1k

diode:d1 2 0 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.

*Simulation Output
**********************************************************
.out plot term 2 vf invfft 2 repeat in "simplediodev.whb"
.out plot term 1 vf term 2 vf sub invfft 2 repeat in "simplediodеresv.whb"

.end
```

Appendix A.2 Full Wave Rectifier Circuit Netlist

```
*** Wave HB Full Wave Rectifier ***

*Simulation options
**********************************************************
.options freq=60

*Simulation Type and setup parameters
**********************************************************
.wavehb n_freqs = 25  fundamental = freq tol=1e-7 n_iter = 10 zref = 800 max_iter = 1000 tol_step=1

*Circuit netlist
**********************************************************
vsource:v1 1 3 f=freq vac=10 phase=-90

resistor:rs 1 2 r=5
resistor:rl 4 0 r=10k

diode:d1 2 4 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d2 3 4 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d3 0 2 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d4 0 3 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.

*Simulation Output
**********************************************************
.out plot term 4 vf invfft 2 repeat in "recvout.whb"
.out plot term 1 vf term 3 vf sub invfft 2 repeat in "recresv.whb"
.out plot term 4 vf in "recvout.whb_p"
.out plot term 1 vf term 3 vf sub in "recresv.whb_p"

.end
```

Appendix A.3 Charge Pump Circuit Netlist

```
***Wave HB Charge Pump Circuit***

*Simulation options
**********************************************************
.options freq=1e6

*Simulation Type and setup parameters
**********************************************************
.wavehb n_freqs=25  fundamental=freq tol=1e-5 n_iter=200 usecaps=0 zref=50 max_iter=1600 tol_step = 0

*Circuit netlist
**********************************************************
vsource:v1 0 1 f=freq vac=10 phase=-90

resistor:rs 1 2 r=1
resistor:rl 0 7 r=1e6

diode:d1 0 4 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d2 4 5 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d3 5 6 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.
diode:d4 6 7 js=5.1e-14 alfa=38.696 jb=1.0e-5 vb=-1.0e50 e=10
+ CT0=1.32767e-15 gama=0.810205 fi=1.27517 afac=38.696 area=271 r0=10.

capacitor:c1 2 4 c=1e-5
capacitor:c2 0 5 c=1e-5
capacitor:c3 4 6 c=1e-5
capacitor:c4 5 7 c=1e-5

*Simulation Output
**********************************************************
.out plot term 7 vf invfft 4 repeat in "chargepumpvout.whb"
.out plot term 1 vf invfft 2 repeat in "chargepumpvoutvin.whb"

.end
```

Appendix A.4 MESFET Amplifier Circuit Netlist

```
*** Wave HB MESFET Amplifier Circuit ***

*Simulation options
***********************************************************
.options f0 = 5.1e9 jupdm=4 output=0 nonlin=4

*Simulation Type and setup parameters
***********************************************************
.wavehb n_freqs=20  fundamental=f0 tol=1e-6 n_iter=200 usecaps=0 zref=50 extrap=1 n_extrap=3 extol=0.01 max_iter=100
tol_step=0

*Circuit netlist
***********************************************************
inductor:l1 1 2 l=1e-9 time_d=0
capacitor:c1 2 3 c=20e-11 time_d=0
inductor:l2 3 7 l=15e-9 time_d=0
resistor:r2 7 8 r=100
mesfetm:m1 3 4 123 idss = 0.06 vp0 = -1.906 gama = -0.015 e = 1.8
+ sl = 0.0676 kg = 1.1 t = 7.0e-12 ss = 1.666e-3 ig0 = 7.13e-6
+ afag = 38.46 r10 = 3.5 kr = 1.111 vbc = 12 ib0 = 7.13e-6 afab = 38.46
+ c10 = 0.42e-12 k1 = 1.282 cf0 = 0.02e-12 kf = 1.282
resistor:rs 123 0 r=1.144
inductor:l3 4 5 l=15e-9 time_d=0
resistor:r3 5 6 r=10
capacitor:cload 4 9 c=20e-12 time_d=0
resistor:rload 9 0 r=50.
vsource:vbias 8 0  vdc = -.4
vsource:vdrain 6 0 vdc = 3.
resistor:rin 11 1 r = 50
vsource:vs 11 0 f = f0 vac = 1.

*Simulation Output
***********************************************************
.out plot term 4 vf term 123 vf sub invfft 4 repeat in "out.vds"
.out plot term 4 vf term 123 vf sub mag in "out.vds.mag"
.out plot element "mesfetm:m1" 1 if invfft 4 repeat in "out.ids"

.end
```

Appendix A.5 Soliton Line Circuit Netlist

```
*** Wave HB MESFET Amplifier Circuit ***

*Simulation options
**********************************************************
* Transim file for NLTL with 24.00 GHz initial Bragg frequency,
* 225.00 GHz final Bragg frequency and 0.952097 tapering rule,
* and 120.00 ps total compression.
.options freq=9.GHz nonlin=4

*Simulation Type and setup parameters
**********************************************************
.wavehb n_freqs=40 fundamental=freq oversample=8 tol=1e-6 zref=50 n_iter=10 tol_step=1 max_iter=1000 usecaps=0

*Circuit netlist
**********************************************************
* For 27dBm input use vac = 14V
vsource:1 201 0 vac = 14. vdc = -6. f = 9e9 phase=90
resistor:rs 201 202 r=50.
* Diode parameters
* From thesis: js=2.24e-12, alfa=21.13
* From Libra netlist: js=51e-15, alfa=default
.model carlos diode ( js=2.24e-12 alfa=21.13 e=10 ct0=1.32767e-15 r0=171.9
+ fi=1.27517 gama=0.810205 jb=1.e-5 vb=-16.)
* Transmission line parameters
.model c_line tlinp4 ( z0mag=75.00 k=7 fscale=1.e10
+ alpha = 59.9 )
* Diodes
diode:d1 101 0 model = "carlos" area=271.64
diode:d2 102 0 model = "carlos" area=258.63
diode:d3 103 0  model = "carlos" area=246.24
diode:d4 104 0  model = "carlos" area=234.45
diode:d5 105 0  model = "carlos" area=223.21
diode:d6 106 0  model = "carlos" area=212.52
diode:d7 107 0  model = "carlos" area=202.34
diode:d8 108 0  model = "carlos" area=192.65
diode:d9 109 0  model = "carlos" area=183.42
diode:d10 110 0  model = "carlos" area=174.63
diode:d11 111 0  model = "carlos" area=166.27
diode:d12 112 0  model = "carlos" area=158.3
diode:d13 113 0  model = "carlos" area=150.72
diode:d14 114 0  model = "carlos" area=143.5
diode:d15 115 0  model = "carlos" area=136.63
diode:d16 116 0  model = "carlos" area=130.08
diode:d17 117 0  model = "carlos" area=123.85
diode:d18 118 0  model = "carlos" area=117.92
diode:d19 119 0  model = "carlos" area=112.27
diode:d20 120 0  model = "carlos" area=106.89
diode:d21 121 0  model = "carlos" area=101.77
diode:d22 122 0  model = "carlos" area=96.89
diode:d23 123 0  model = "carlos" area=92.25
diode:d24 124 0  model = "carlos" area=87.83
diode:d25 125 0  model = "carlos" area=83.63
diode:d26 126 0  model = "carlos" area=79.62
diode:d27 127 0  model = "carlos" area=75.81
diode:d28 128 0  model = "carlos" area=72.18
diode:d29 129 0  model = "carlos" area=68.72
diode:d30 130 0  model = "carlos" area=65.43
diode:d31 131 0  model = "carlos" area=62.29
diode:d32 132 0  model = "carlos" area=59.31
diode:d33 133 0  model = "carlos" area=56.47
diode:d34 134 0  model = "carlos" area=53.76
diode:d35 135 0  model = "carlos" area=51.19
diode:d36 136 0  model = "carlos" area=48.73
```

```
diode:d37 137 0  model = "carlos" area=46.4
diode:d38 138 0  model = "carlos" area=44.18
diode:d39 139 0  model = "carlos" area=42.06
diode:d40 140 0  model = "carlos" area=40.05
diode:d41 141 0  model = "carlos" area=38.13
diode:d42 142 0  model = "carlos" area=36.3
diode:d43 143 0  model = "carlos" area=34.56
diode:d44 144 0  model = "carlos" area=32.91
diode:d45 145 0  model = "carlos" area=31.33
diode:d46 146 0  model = "carlos" area=29.83
diode:d47 147 0  model = "carlos" area=28.4
* Parasitic inductors
inductor:i1  1 101 l=21.8pH
inductor:i2  2 102 l=21.8pH
inductor:i3  3 103 l=21.8pH
inductor:i4  4 104 l=21.8pH
inductor:i5  5 105 l=21.8pH
inductor:i6  6 106 l=21.8pH
inductor:i7  7 107 l=21.8pH
inductor:i8  8 108 l=21.8pH
inductor:i9  9 109 l=21.8pH
inductor:i10 10 110 l=21.8pH
inductor:i11 11 111 l=21.8pH
inductor:i12 12 112 l=21.8pH
inductor:i13 13 113 l=21.8pH
inductor:i14 14 114 l=21.8pH
inductor:i15 15 115 l=21.8pH
inductor:i16 16 116 l=21.8pH
inductor:i17 17 117 l=21.8pH
inductor:i18 18 118 l=21.8pH
inductor:i19 19 119 l=21.8pH
inductor:i20 20 120 l=21.8pH
inductor:i21 21 121 l=21.8pH
inductor:i22 22 122 l=21.8pH
inductor:i23 23 123 l=21.8pH
inductor:i24 24 124 l=21.8pH
inductor:i25 25 125 l=21.8pH
inductor:i26 26 126 l=21.8pH
inductor:i27 27 127 l=21.8pH
inductor:i28 28 128 l=21.8pH
inductor:i29 29 129 l=21.8pH
inductor:i30 30 130 l=21.8pH
inductor:i31 31 131 l=21.8pH
inductor:i32 32 132 l=21.8pH
inductor:i33 33 133 l=21.8pH
inductor:i34 34 134 l=21.8pH
inductor:i35 35 135 l=21.8pH
inductor:i36 36 136 l=21.8pH
inductor:i37 37 137 l=21.8pH
inductor:i38 38 138 l=21.8pH
inductor:i39 39 139 l=21.8pH
inductor:i40 40 140 l=21.8pH
inductor:i41 41 141 l=21.8pH
inductor:i42 42 142 l=21.8pH
inductor:i43 43 143 l=21.8pH
inductor:i44 44 144 l=21.8pH
inductor:i45 45 145 l=21.8pH
inductor:i46 46 146 l=21.8pH
inductor:i47 47 147 l=21.8pH
* Transmission lines
tlinp4:t0 202 0 1 0 model = "c_line" length=501.29u
tlinp4:t1  1 0 2 0  model = "c_line" length=978.57u
tlinp4:t2  2 0 3 0 model = "c_line" length=931.69u
tlinp4:t3  3 0 4 0 model = "c_line" length=887.06u
tlinp4:t4  4 0 5 0  model = "c_line" length=844.57u
tlinp4:t5  5 0 6 0  model = "c_line" length=804.11u
tlinp4:t6  6 0 7 0  model = "c_line" length=765.59u
```

```
tlinp4:t7  7 0 8 0  model = "c_line" length=728.92u
tlinp4:t8  8 0 9 0  model = "c_line" length=694.00u
tlinp4:t9  9 0 10 0 model = "c_line" length=660.75u
tlinp4:t10  10 0 11 0 model = "c_line" length=629.10u
tlinp4:t11  11 0 12 0 model = "c_line" length=598.97u
tlinp4:t12  12 0 13 0 model = "c_line" length=570.27u
tlinp4:t13  13 0 14 0 model = "c_line" length=542.96u
tlinp4:t14  14 0 15 0 model = "c_line" length=516.95u
tlinp4:t15  15 0 16 0 model = "c_line" length=492.18u
tlinp4:t16  16 0 17 0 model = "c_line" length=468.61u
tlinp4:t17  17 0 18 0 model = "c_line" length=446.16u
tlinp4:t18  18 0 19 0 model = "c_line" length=424.79u
tlinp4:t19  19 0 20 0 model = "c_line" length=404.44u
tlinp4:t20  20 0 21 0 model = "c_line" length=385.06u
tlinp4:t21  21 0 22 0 model = "c_line" length=366.62u
tlinp4:t22  22 0 23 0 model = "c_line" length=349.05u
tlinp4:t23  23 0 24 0 model = "c_line" length=332.33u
tlinp4:t24  24 0 25 0 model = "c_line" length=316.41u
tlinp4:t25  25 0 26 0 model = "c_line" length=301.26u
tlinp4:t26  26 0 27 0 model = "c_line" length=286.83u
tlinp4:t27  27 0 28 0 model = "c_line" length=273.09u
tlinp4:t28  28 0 29 0 model = "c_line" length=260.00u
tlinp4:t29  29 0 30 0 model = "c_line" length=247.55u
tlinp4:t30  30 0 31 0 model = "c_line" length=235.69u
tlinp4:t31  31 0 32 0 model = "c_line" length=224.40u
tlinp4:t32  32 0 33 0 model = "c_line" length=213.65u
tlinp4:t33  33 0 34 0 model = "c_line" length=203.42u
tlinp4:t34  34 0 35 0 model = "c_line" length=193.67u
tlinp4:t35  35 0 36 0 model = "c_line" length=184.39u
tlinp4:t36  36 0 37 0 model = "c_line" length=175.56u
tlinp4:t37  37 0 38 0 model = "c_line" length=167.15u
tlinp4:t38  38 0 39 0 model = "c_line" length=159.14u
tlinp4:t39  39 0 40 0 model = "c_line" length=151.52u
tlinp4:t40  40 0 41 0 model = "c_line" length=144.26u
tlinp4:t41  41 0 42 0 model = "c_line" length=137.35u
tlinp4:t42  42 0 43 0 model = "c_line" length=130.77u
tlinp4:t43  43 0 44 0 model = "c_line" length=124.51u
tlinp4:t44  44 0 45 0 model = "c_line" length=118.54u
tlinp4:t45  45 0 46 0 model = "c_line" length=112.86u
tlinp4:t46  46 0 47 0 model = "c_line" length=107.46u
tlinp4:t47  47 0 48 0 model = "c_line" length=52.41u
resistor:rl   48  0  r=50.

*Simulation Output
************************************************************
.out plot term 48 vf in "vout.complex"
.out plot term 1 vf in "vdiode1.complex"
.out plot term 47 vf in "vdiode47.complex"
.out plot term 101 vf invfft 5 repeat in "vdiode1.wave"
.out plot term 122 vf invfft 5 repeat in "vdiode22.wave"
.out plot term 147 vf invfft 5 repeat in "vdiode47.wave"
.out plot term 48 vf invfft 5 repeat in "vout.wave"
.out plot term 202 vf invfft 5 repeat in "vin.wave"
.out plot element "diode:d1" 0 if invfft 5 repeat in "diode1.current"
.out plot element "diode:d47" 0 if invfft 5 repeat in "diode47.current"

.end
```

Appendix A.5 Multiple Soliton Line Circuit Netlist

```
*** Wave HB MESFET Amplifier Circuit ***

* Transim file for NLTL with 24.00 GHz initial Bragg frequency,
* 225.00 GHz final Bragg frequency and 0.952097 tapering rule,
* and 120.00 ps total compression.

*Simulation options
**********************************************************
.options freq=9.GHz nonlin=4  ftol = 1e-5

*Simulation Type and setup parameters
**********************************************************
.wavehb n_freqs = 40 fundamental = freq oversample= 8 tol=1e-2 usecaps = 1 zref=50 n_iter=10


*Circuit netlist
**********************************************************
* For 27dBm input use vac = 14V
vsource:1 201 0 vac = 14. vdc = -6. f = 9.GHz phase=90 tr=.1e-9
resistor:rs1 201 1 r=50.
vsource:2 202 0 vac = 14. vdc = -6. f = 9.GHz phase=-90 tr=.1e-9
resistor:rs2 202 4 r=50.
xline1 1 2 0 nltline
xline2 4 5 0 nltline
resistor:rl1 2 0 r=100.
resistor:rl2 5 0 r=100.
resistor:rl3 7 0 r=100.
resistor:rp1 2 5 r=80.
resistor:rp2 5 7 r=80.
*****************************************************
* Definition of the Non-linear Tr Line model
* Input node = 202
* Output node = 48
* Common node = "GND1" (GND)
*****************************************************
.subckt nltline 202 48 "GND1"
* Diode parameters
* From thesis: js=2.24e-12, alfa=21.13
* From Libra netlist: js=51e-15, alfa=default
.model carlos diode ( js=2.24e-12 alfa=21.13 e=10 ct0=1.32767e-15 r0=171.9
+ fi=1.27517 gama=0.810205 jb=1.e-5 vb=-16. )
* Transmission line parameters
.model c_line tlinp4 ( z0mag=75.00 k=7 fscale=10.e9 alpha = 59.9
+ nsect = 20 fopt=10e9)
* Diodes
diode:d1 101 "GND1" model = "carlos" area=271.64
diode:d2 102 "GND1" model = "carlos" area=258.63
diode:d3 103 "GND1"  model = "carlos" area=246.24
diode:d4 104 "GND1"  model = "carlos" area=234.45
diode:d5 105 "GND1"  model = "carlos" area=223.21
diode:d6 106 "GND1"  model = "carlos" area=212.52
diode:d7 107 "GND1"  model = "carlos" area=202.34
diode:d8 108 "GND1"  model = "carlos" area=192.65
diode:d9 109 "GND1"  model = "carlos" area=183.42
diode:d10 110 "GND1"  model = "carlos" area=174.63
diode:d11 111 "GND1"  model = "carlos" area=166.27
diode:d12 112 "GND1"  model = "carlos" area=158.3
diode:d13 113 "GND1"  model = "carlos" area=150.72
diode:d14 114 "GND1"  model = "carlos" area=143.5
diode:d15 115 "GND1"  model = "carlos" area=136.63
diode:d16 116 "GND1"  model = "carlos" area=130.08
diode:d17 117 "GND1"  model = "carlos" area=123.85
diode:d18 118 "GND1"  model = "carlos" area=117.92
```

```
diode:d19 119 "GND1"  model = "carlos" area=112.27
diode:d20 120 "GND1"  model = "carlos" area=106.89
diode:d21 121 "GND1"  model = "carlos" area=101.77
diode:d22 122 "GND1"  model = "carlos" area=96.89
diode:d23 123 "GND1"  model = "carlos" area=92.25
diode:d24 124 "GND1"  model = "carlos" area=87.83
diode:d25 125 "GND1"  model = "carlos" area=83.63
diode:d26 126 "GND1"  model = "carlos" area=79.62
diode:d27 127 "GND1"  model = "carlos" area=75.81
diode:d28 128 "GND1"  model = "carlos" area=72.18
diode:d29 129 "GND1"  model = "carlos" area=68.72
diode:d30 130 "GND1"  model = "carlos" area=65.43
diode:d31 131 "GND1"  model = "carlos" area=62.29
diode:d32 132 "GND1"  model = "carlos" area=59.31
diode:d33 133 "GND1"  model = "carlos" area=56.47
diode:d34 134 "GND1"  model = "carlos" area=53.76
diode:d35 135 "GND1"  model = "carlos" area=51.19
diode:d36 136 "GND1"  model = "carlos" area=48.73
diode:d37 137 "GND1"  model = "carlos" area=46.4
diode:d38 138 "GND1"  model = "carlos" area=44.18
diode:d39 139 "GND1"  model = "carlos" area=42.06
diode:d40 140 "GND1"  model = "carlos" area=40.05
diode:d41 141 "GND1"  model = "carlos" area=38.13
diode:d42 142 "GND1"  model = "carlos" area=36.3
diode:d43 143 "GND1"  model = "carlos" area=34.56
diode:d44 144 "GND1"  model = "carlos" area=32.91
diode:d45 145 "GND1"  model = "carlos" area=31.33
diode:d46 146 "GND1"  model = "carlos" area=29.83
diode:d47 147 "GND1"  model = "carlos" area=28.4
* Parasitic inductors
inductor:i1  1 101 l=21.8pH
inductor:i2  2 102 l=21.8pH
inductor:i3  3 103 l=21.8pH
inductor:i4  4 104 l=21.8pH
inductor:i5  5 105 l=21.8pH
inductor:i6  6 106 l=21.8pH
inductor:i7  7 107 l=21.8pH
inductor:i8  8 108 l=21.8pH
inductor:i9  9 109 l=21.8pH
inductor:i10 10 110 l=21.8pH
inductor:i11 11 111 l=21.8pH
inductor:i12 12 112 l=21.8pH
inductor:i13 13 113 l=21.8pH
inductor:i14 14 114 l=21.8pH
inductor:i15 15 115 l=21.8pH
inductor:i16 16 116 l=21.8pH
inductor:i17 17 117 l=21.8pH
inductor:i18 18 118 l=21.8pH
inductor:i19 19 119 l=21.8pH
inductor:i20 20 120 l=21.8pH
inductor:i21 21 121 l=21.8pH
inductor:i22 22 122 l=21.8pH
inductor:i23 23 123 l=21.8pH
inductor:i24 24 124 l=21.8pH
inductor:i25 25 125 l=21.8pH
inductor:i26 26 126 l=21.8pH
inductor:i27 27 127 l=21.8pH
inductor:i28 28 128 l=21.8pH
inductor:i29 29 129 l=21.8pH
inductor:i30 30 130 l=21.8pH
inductor:i31 31 131 l=21.8pH
inductor:i32 32 132 l=21.8pH
inductor:i33 33 133 l=21.8pH
inductor:i34 34 134 l=21.8pH
inductor:i35 35 135 l=21.8pH
inductor:i36 36 136 l=21.8pH
inductor:i37 37 137 l=21.8pH
```

```
inductor:i38 38 138 l=21.8pH
inductor:i39 39 139 l=21.8pH
inductor:i40 40 140 l=21.8pH
inductor:i41 41 141 l=21.8pH
inductor:i42 42 142 l=21.8pH
inductor:i43 43 143 l=21.8pH
inductor:i44 44 144 l=21.8pH
inductor:i45 45 145 l=21.8pH
inductor:i46 46 146 l=21.8pH
inductor:i47 47 147 l=21.8pH
* Transmission lines
tlinp4:t0 202 "GND1" 1 "GND1" model = "c_line" length=501.29u
tlinp4:t1  1 "GND1" 2 "GND1"  model = "c_line" length=978.57u
tlinp4:t2  2 "GND1" 3 "GND1" model = "c_line" length=931.69u
tlinp4:t3  3 "GND1" 4 "GND1" model = "c_line" length=887.06u
tlinp4:t4  4 "GND1" 5 "GND1"  model = "c_line" length=844.57u
tlinp4:t5  5 "GND1" 6 "GND1"  model = "c_line" length=804.11u
tlinp4:t6  6 "GND1" 7 "GND1"  model = "c_line" length=765.59u
tlinp4:t7  7 "GND1" 8 "GND1"  model = "c_line" length=728.92u
tlinp4:t8  8 "GND1" 9 "GND1"  model = "c_line" length=694.00u
tlinp4:t9  9 "GND1" 10 "GND1" model = "c_line" length=660.75u
tlinp4:t10  10 "GND1" 11 "GND1" model = "c_line" length=629.10u
tlinp4:t11  11 "GND1" 12 "GND1" model = "c_line" length=598.97u
tlinp4:t12  12 "GND1" 13 "GND1" model = "c_line" length=570.27u
tlinp4:t13  13 "GND1" 14 "GND1" model = "c_line" length=542.96u
tlinp4:t14  14 "GND1" 15 "GND1" model = "c_line" length=516.95u
tlinp4:t15  15 "GND1" 16 "GND1" model = "c_line" length=492.18u
tlinp4:t16  16 "GND1" 17 "GND1" model = "c_line" length=468.61u
tlinp4:t17  17 "GND1" 18 "GND1" model = "c_line" length=446.16u
tlinp4:t18  18 "GND1" 19 "GND1" model = "c_line" length=424.79u
tlinp4:t19  19 "GND1" 20 "GND1" model = "c_line" length=404.44u
tlinp4:t20  20 "GND1" 21 "GND1" model = "c_line" length=385.06u
tlinp4:t21  21 "GND1" 22 "GND1" model = "c_line" length=366.62u
tlinp4:t22  22 "GND1" 23 "GND1" model = "c_line" length=349.05u
tlinp4:t23  23 "GND1" 24 "GND1" model = "c_line" length=332.33u
tlinp4:t24  24 "GND1" 25 "GND1" model = "c_line" length=316.41u
tlinp4:t25  25 "GND1" 26 "GND1" model = "c_line" length=301.26u
tlinp4:t26  26 "GND1" 27 "GND1" model = "c_line" length=286.83u
tlinp4:t27  27 "GND1" 28 "GND1" model = "c_line" length=273.09u
tlinp4:t28  28 "GND1" 29 "GND1" model = "c_line" length=260.00u
tlinp4:t29  29 "GND1" 30 "GND1" model = "c_line" length=247.55u
tlinp4:t30  30 "GND1" 31 "GND1" model = "c_line" length=235.69u
tlinp4:t31  31 "GND1" 32 "GND1" model = "c_line" length=224.40u
tlinp4:t32  32 "GND1" 33 "GND1" model = "c_line" length=213.65u
tlinp4:t33  33 "GND1" 34 "GND1" model = "c_line" length=203.42u
tlinp4:t34  34 "GND1" 35 "GND1" model = "c_line" length=193.67u
tlinp4:t35  35 "GND1" 36 "GND1" model = "c_line" length=184.39u
tlinp4:t36  36 "GND1" 37 "GND1" model = "c_line" length=175.56u
tlinp4:t37  37 "GND1" 38 "GND1" model = "c_line" length=167.15u
tlinp4:t38  38 "GND1" 39 "GND1" model = "c_line" length=159.14u
tlinp4:t39  39 "GND1" 40 "GND1" model = "c_line" length=151.52u
tlinp4:t40  40 "GND1" 41 "GND1" model = "c_line" length=144.26u
tlinp4:t41  41 "GND1" 42 "GND1" model = "c_line" length=137.35u
tlinp4:t42  42 "GND1" 43 "GND1" model = "c_line" length=130.77u
tlinp4:t43  43 "GND1" 44 "GND1" model = "c_line" length=124.51u
tlinp4:t44  44 "GND1" 45 "GND1" model = "c_line" length=118.54u
tlinp4:t45  45 "GND1" 46 "GND1" model = "c_line" length=112.86u
tlinp4:t46  46 "GND1" 47 "GND1" model = "c_line" length=107.46u
tlinp4:t47  47 "GND1" 48 "GND1" model = "c_line" length=52.41u
.ends
********************************************************
*Simulation Output
********************************************************
.out plot term 2 vf invfft 5 repeat in "vdiode22.wave"
.out plot term 2 vf invfft 5 repeat in "vdiode47.wave"

.end
```

Appendix B

Appendix B.1 Extrapolation Results of Simple Diode Circuit

To try and improve the convergence performance further the use of extrapolation is tested on both the reflected wave and the port voltage while using capacitors.

	Diode	Diode
Number Harmonics	15	15
Solution Tolerance	1.00E-008	1.00E-008
Tolerance Stepping		
Max Port Iterations	100	100
Zref	800	800
Update Scaling Factor	0.3	0.3
Max NL Iterations	1000	1000
Use Caps	1	0
Conductance Value	1/250	NA
Max Cap Iterations	1000	NA
Iterative time	0	NA
Use Extrapolation	0	1
Extrapolation Start Point	NA	1.00E-001
Number of Extrapolation Samples	NA	3
Use Cap Extrapolation	1	NA
Cap Extrapolation Start Point	1.00E-005	NA
Number of Cap Extrapolation Samples	3	NA

Table B.1 Simple Diode Extrapolation Setup

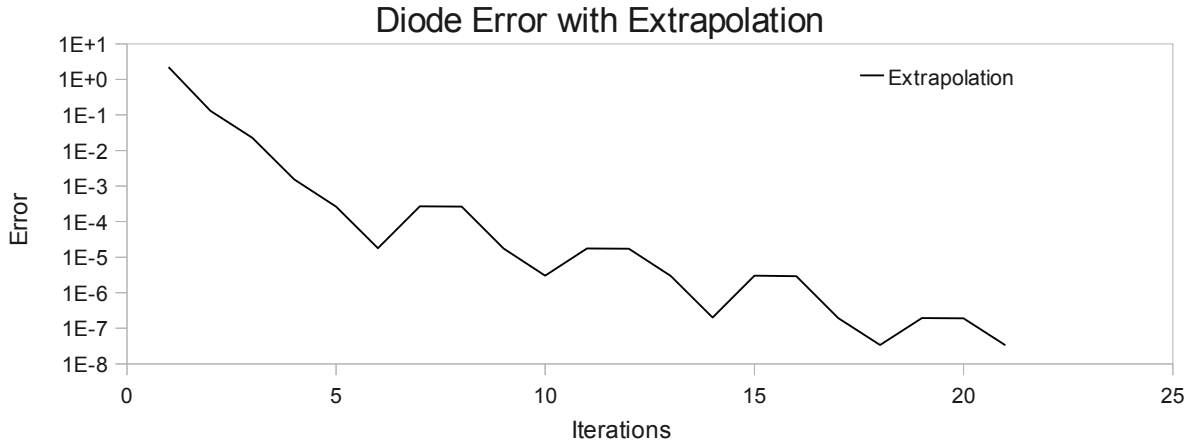

Fig. B.1 Extrapolation on Reflected Wave

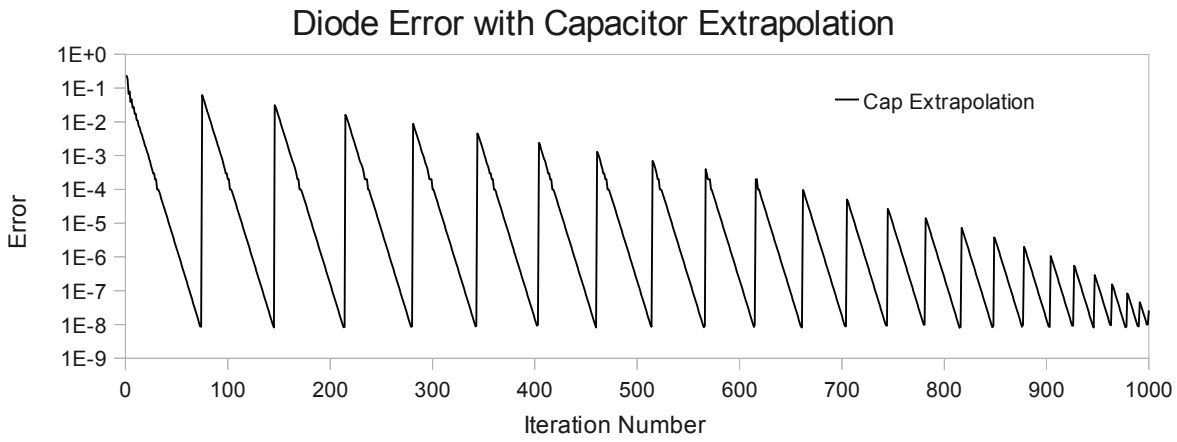

Fig. B.2 Capacitor Extrapolation

The residual of the error after performing an extrapolation increases dramatically and then slowly decreases until another extrapolation is performed. The simulation was stopped by the maximum number of iterations threshold defined in the simulation setup. The trend appears to be converging to solution when the iteration threshold is hit.

Appendix B.2 Extrapolation Results of Full Wave Rectifier Circuit

	Rectifier	Rectifier
Number Harmonics	25	25
Solution Tolerance	1.00E-007	1.00E-007
Tolerance Stepping	0	0
Max Port Iterations	200	200
Zref	800	800
Update Scaling Factor	0.3	0.3
Max NL Iterations	1000	1000
Use Caps	1	0
Conductance Value	1/250	NA
Max Cap Iterations	300	NA
Iterative time	0	NA
Use Extrapolation	0	1
Extrapolation Start Point	NA	1.00E-003
Number of Extrapolation Samples	NA	5
Use Cap Extrapolation	1	NA
Cap Extrapolation Start Point	1.00E-005	NA
Number of Cap Extrapolation Samples	15	NA

Table B.2 Full Wave Rectifier Extrapolation Setup

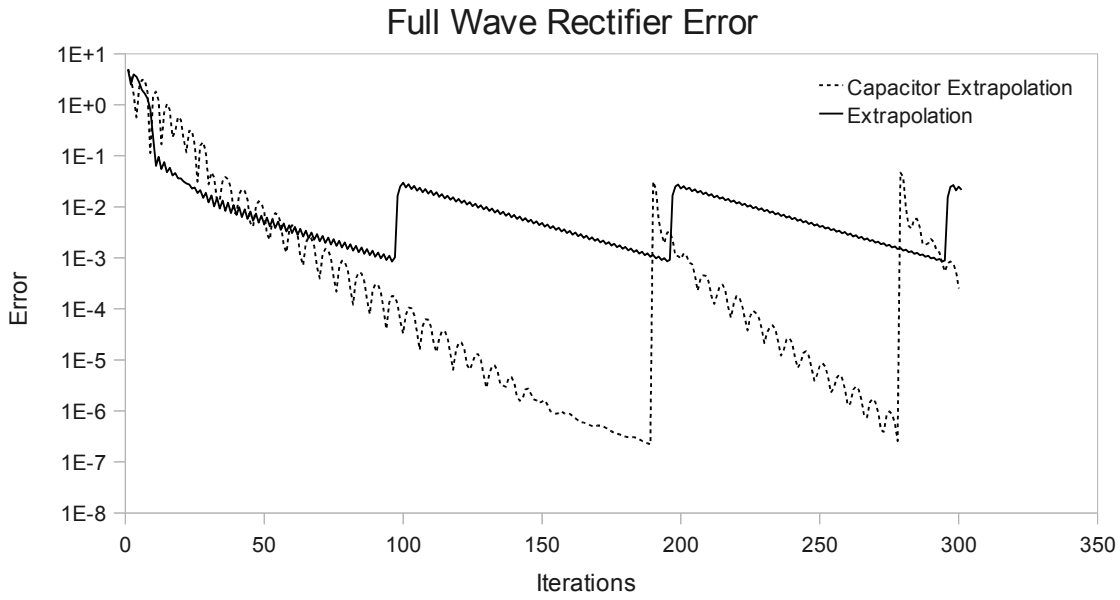

Fig. B.3 Capacitor and Reflected Wave Extrapolation

Appendix B.3 Extrapolation Results of Charge Pump Circuit

	Charge Pump
Number Harmonics	25
Solution Tolerance	1.00E-005
Tolerance Stepping	1
Max Port Iterations	200
Zref	50
Update Scaling Factor	0.3
Max NL Iterations	1600
Use Caps	0
Conductance Value	NA
Max Cap Iterations	NA
Iterative time	NA
Use Extrapolation	1
Extrapolation Start Point	1.00E-002

Table B.3 Charge Pump Extrapolation Setup

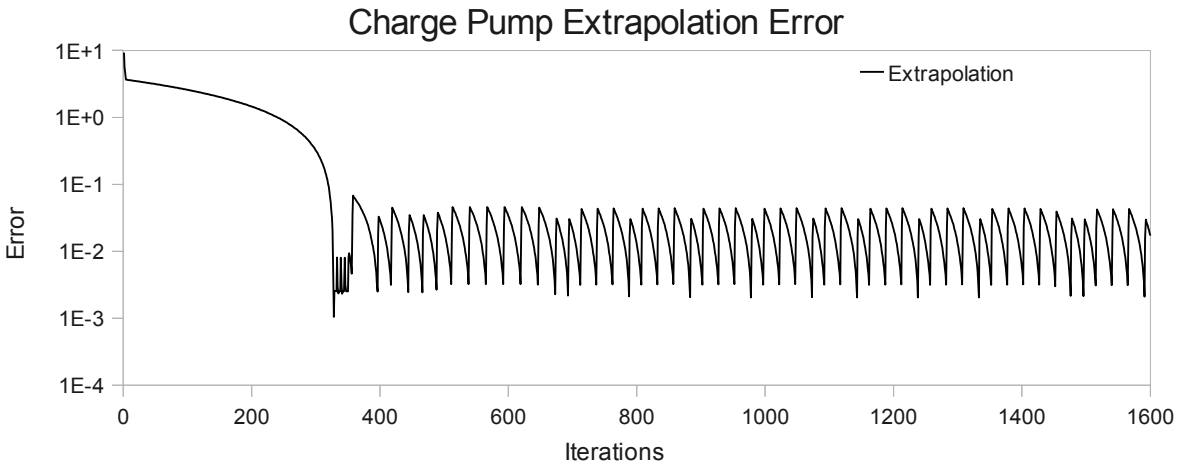

Fig. B.4 Extrapolation on Reflected Wave

Appendix B.4 Extrapolation Results of MESFET Amplifier Circuit

	MESFET
Number Harmonics	25
Solution Tolerance	1.00E-006
Tolerance Stepping	0
Max Port Iterations	200
Zref	50
Update Scaling Factor	0.3
Max NL Iterations	100
Use Extrapolation	1
Extrapolation Start Point	5.00E-001
Number of Extrapolation Samples	3

Table B.4 MESFET Amplifier Extrapolation Setup

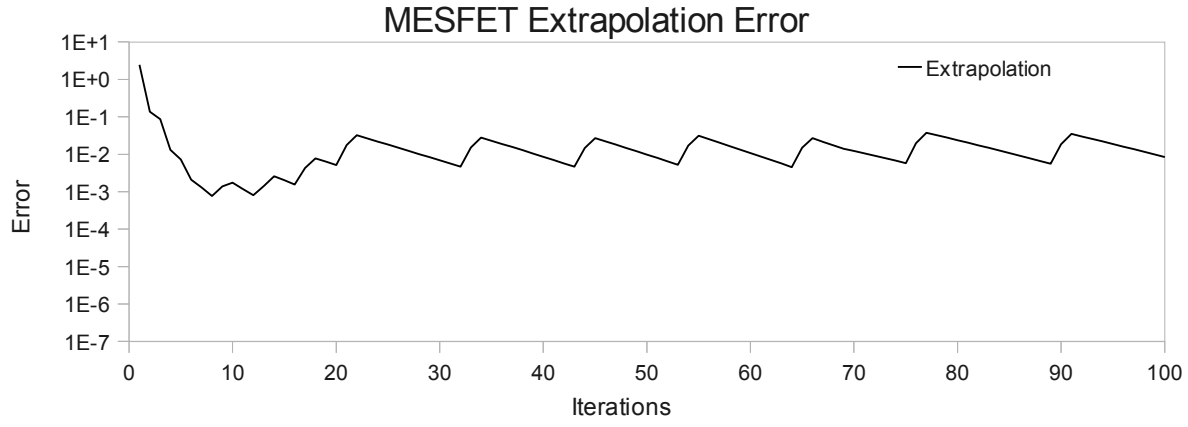

Fig. B.5 Extrapolation on Reflected Wave

References

[1] C. Christoffersen and M. Steer, “State Variable Harmonic Balance Simulation of a Quasi-Optical Power Combing System”, North Carolina State University, Masters Thesis, 1998.

[2] S. Mass, “Nonlinear Microwave and RF Circuits”, Second Edition, CRC Press, 2003.

[3] A. R. Kerr, “A Technique for Determining the Local Oscillator Waveforms in a Microwave Mixer”, IEEE Trans. On Microwave Theory and Techniques, October 1974, pp. 828-831.

[4] R. G. Hicks and P. J. Khan, “Numerical Analysis of Nonlinear Solid-State Device Excitation in Microwave Circuits”, IEEE Trans. On Microwave Theory and Techniques, vol. Mtt-30, March 1982, pp. 251-259.

[5] V. Rizzoli, F. Mastri, D. Masotti and F. Sgallari, “Scattering-Matrix Based Inexact-Newton Harmonic Balance with Incomplete Preconditioning”, European Microwave conference, October 2000, pp. 1-4.

[6] M. S. Nakhla, J. Vlach, “A Piecewise Harmonic Balance Technique for Determination of Periodic Response of Nonlinear Systems”, IEEE Trans. on Circuits and Systems, Vol CAS-23, No. 2, Feb 1976.

[7] S. Rao, “Applied Numerical Methods for Engineers and Scientists”, Prentice Hall, Upper-Saddle River, New Jersey, 2002.

[8] D. Pozar, “Microwave Engineering 3rd Edition”, John Wiley & Sons, Hoboken, New Jersey, 2005.

[9] V. Borich, J. East and G. Haddad, “A Fixed Point Harmonic Balance Approach For Circuit Simulation Under Modulated Carrier Excitation”, Proc. Of the IEEE International Symp. On Circuits and Systems, vol. 6, June 1999, pp. 182-188.

[10] C. Christoffersen, “Global Modeling of Nonlinear Microwave Circuits”, North Carolina State University, PhD. Dissertation, 2000.

[11] F. Veerse, “Efficient Iterative Time Preconditioners for Harmonic Balance RF Circuit simulations”, Computer Aided Design, 2003 International Conference, Nov. 9-13, 2003, pg 251-254

[12] G. Tait, “Efficient Solution Method for Unified NonlInear Microwave Circuit and Numerical Solid-State Device Simulation”, IEEE Microwave and Guided Wave Letter, Vol. 4, No.12, Dec. 1994.

[13] P. Blakey and S. Bates “Convergence Properties of Fixed-Point Harmonic balance Algorithms”, IEEE International Symp. on Signals, Systems and Electronics, Aug. 2007.

[14] V. Rizzoli, F. Mastri, C. Cecchetti and F. Sgallari, “Fast and Robust Inexact Netwon Approach to the Harmonic Balance Analysis of Nonlinear Microwave Circuits”, IEEE Microwave and Guided Wave Letter, Vol. 7, No.10, Dec. 1997.

[15] V. Rizzoli, A. Lipparini, A. Costanzo, F. Mastri, C. Cecchetti, A. Neri and D. Masotti, “State-of-the-Art Harmonic Balance Simulation of Forced Nonlinear Microwave Circuits by the Piecewise Technique”, IEEE Trans. On Microwave Theory and Techniques, Vol. 40, No. 1, Jan. 1992.

[16] M. Honkala, “Parallel Hierarchal DC Analysis”, Thesis for Degree of Licentiate of Science in Technology, Helsinki University of Technology, March 2002.

[17] N. Soveiko, M. Nakhla and R. Achar, “Comparsion Study of Performance of Parallel Steady State Solvers on Different Computer Architechtures”, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol. 29, No. 1, Jan. 2010.

[18] B. Steer, “Global Modeling of Spatially Distributed Microwave and Millimeter-Wave Systems”, IEEE Trans. On Microwave Theory and Techniques, Vol. 47 No. 6, June 1999.

[19] D. Smith, W. Ford and A. Sidi, “Extrapolation Methods for Vector Sequences”, SAIM Review, Vol. 29, No. 2, June 1987.

[20] C. Christoffersen, “Transient Analysis of Nonlinear Circuits Based on Waves”, Lakehead University, 2008.

[21] J. Roychowdhury, “Analyzing Circuits with Widely Separated Time Scales Using Numerical PDE Methods”, IEEE Trans. on Circuits and Systems, Vol 48, No. 5, May 2001.

[22] J. Siek, “A Modern Framework for Portable High Performance Numerical Linear Algebra”, University of Notre Dame, PhD. Dissertation, 1999.

[23] E. Lelarasme, “The Waveform Relaxation Method for the Time-Domain Analysis of Large Scale Integrated Circuits: Theory and Applications”, Ph.D. dissertation, Dept. of EECS, Univ. of California, Berkele, 1982.

[24] http://www.netlib.org/lapack/

[25] http://www.gnu.org/software/octave/doc/interpreter/

[26] L. Nagel and D. Pederson, “SPICE (Simulation Program with Integrated Circuit Emphasis)”, Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973.

[27] M. Kabir, “Nonlinear Transient Analysis Based on Power Waves and State Variables”, Masters Thesis, Lakehead University, May 2010.

[28] H. Shi, C. Domier and N. Luhmann, “A Monolithic Nonlinear Transmission Line System for the Experimental Study of Lattice Solutions”, Journal of Applied Physic, Vol. 4, 1995.

[29] M. J. W. Rodwell, M. Kamegawa, R. Yu, M. Case, E. Carman and K. S. Giboney, “GaAs Nonlinear Transmission Lines for Picosecond Pulse Generation and Millimeter-wave Sampling”, IEEE Trans. on Microwave Theory and Tech., Vol. 39, Issue-7, 1991.

[30] G. Dahlquist and A. Bjorck, “Numerical Methods”, Prentice Hall, 1974.

[31] http://en.wikipedia.org/wiki/Gradient_descent